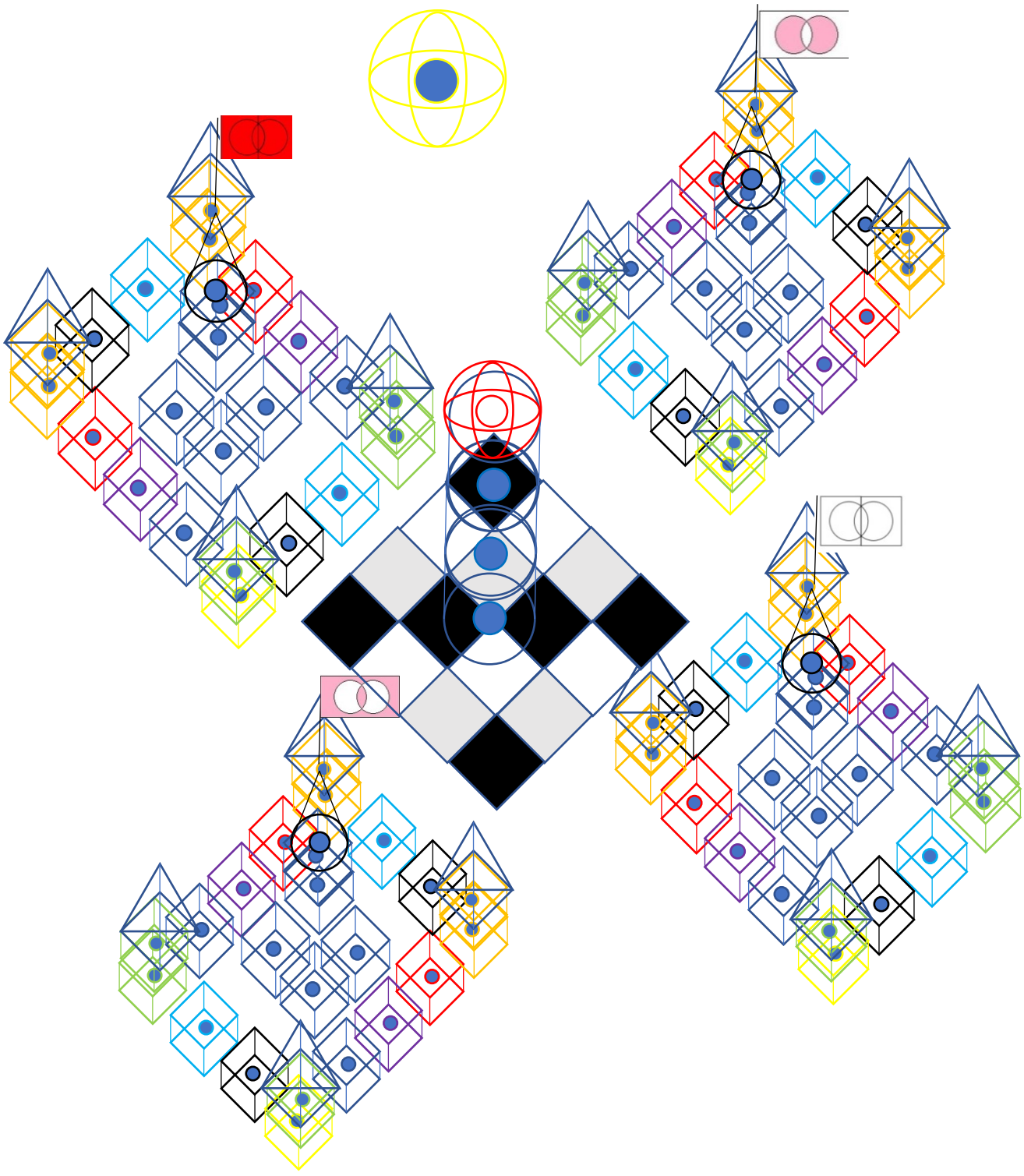
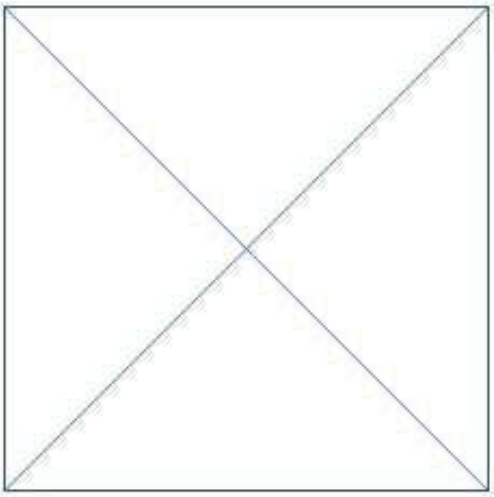
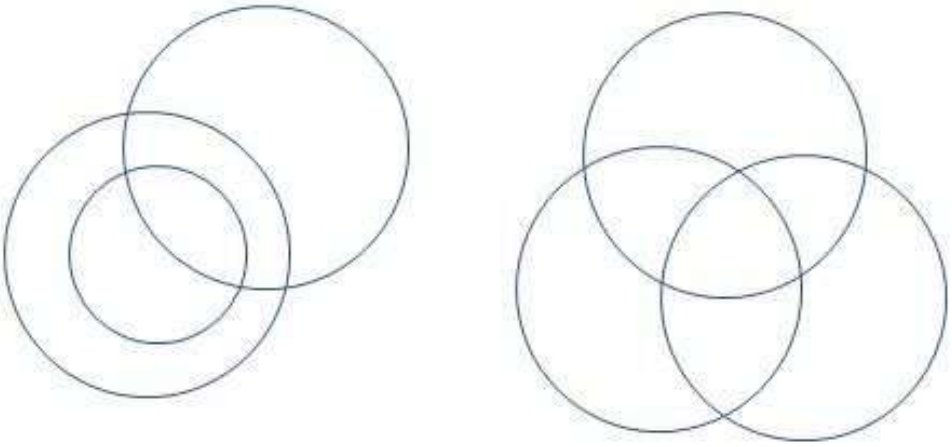
Modernization of Classic Logic



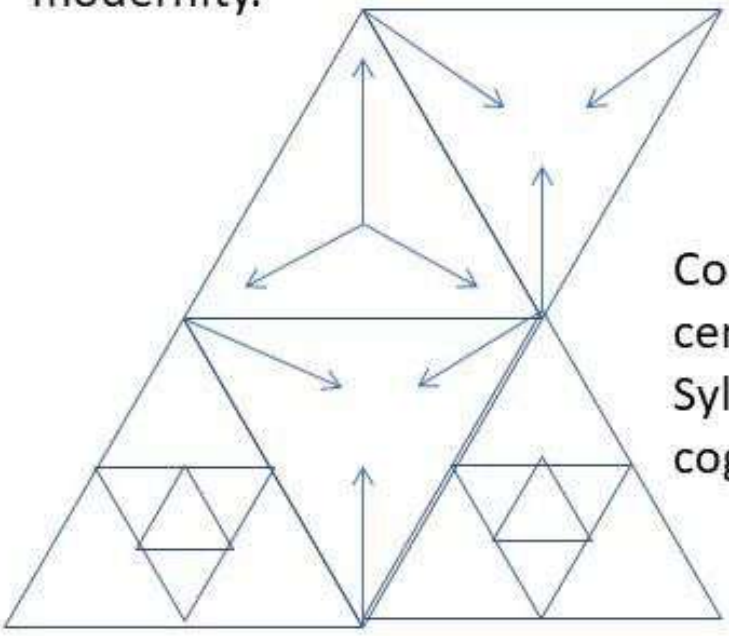
Model of Reasoning
Logic of the Great, Logic of the Wise
Fedorchenko Mikhail Valerevich



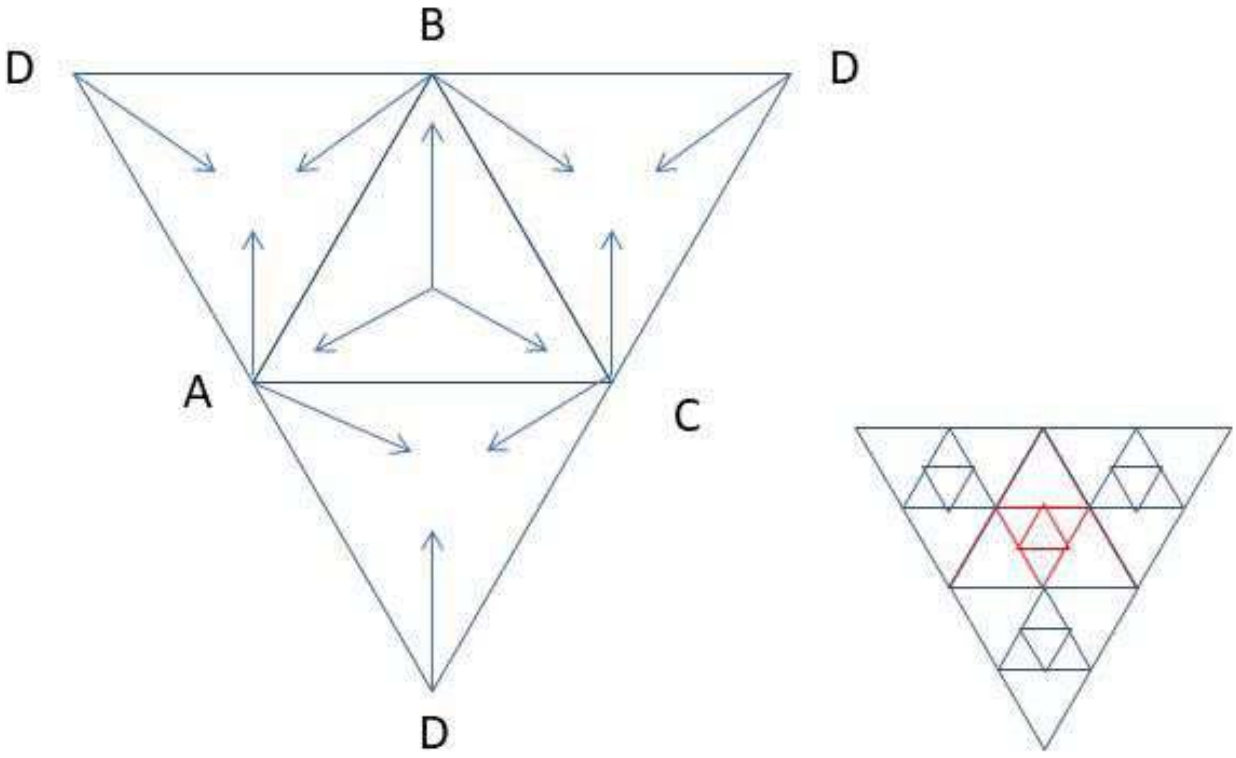
Early-late medieval computing.

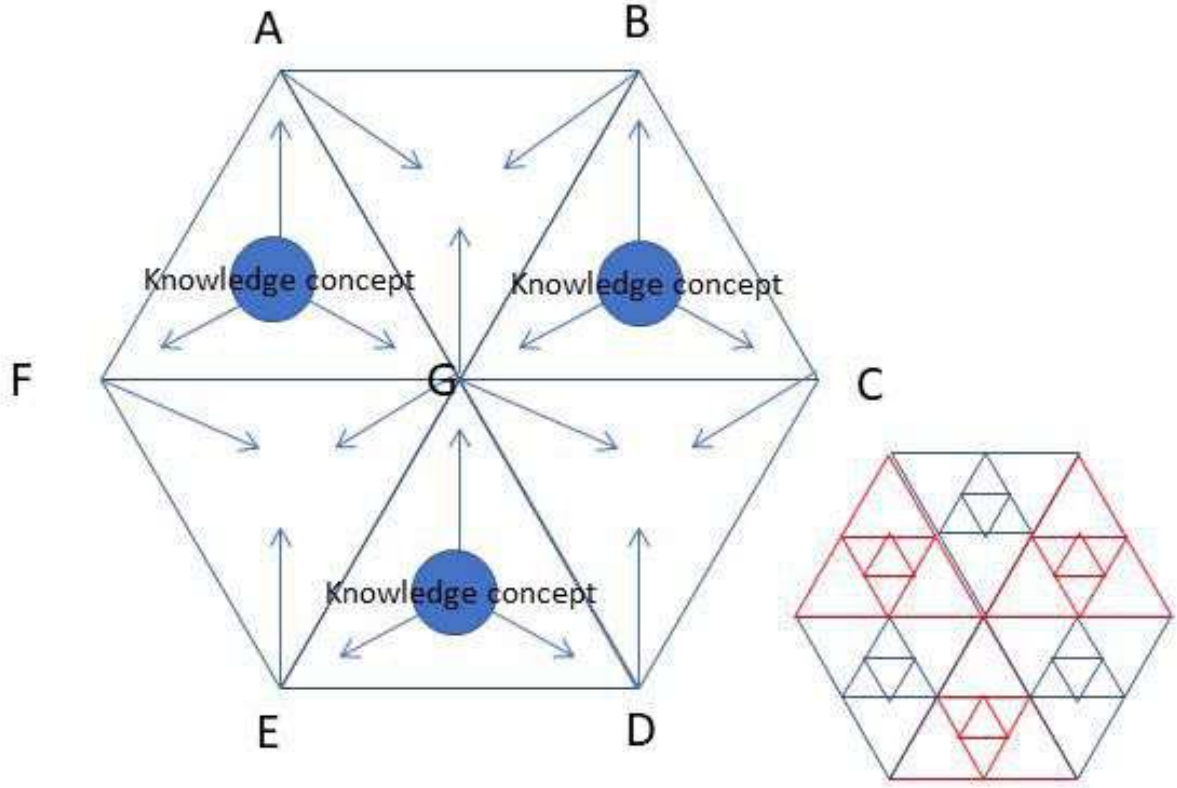


Computing from 18th century-1923-modernity.



Computing from the 21st century,2017 is the future. Syllogistics intellectum cognitus

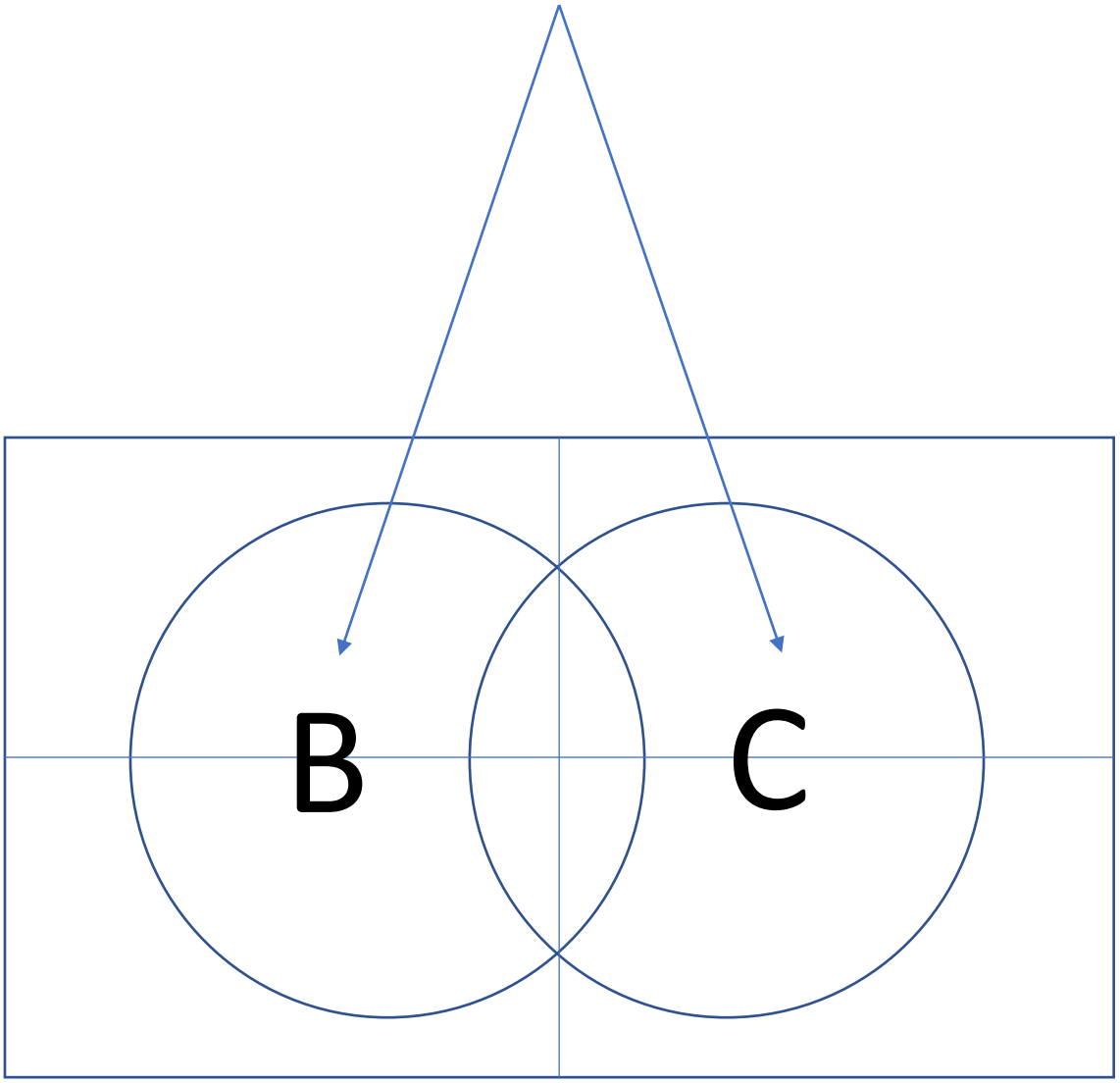




Федорченко Михаил Валерьевич

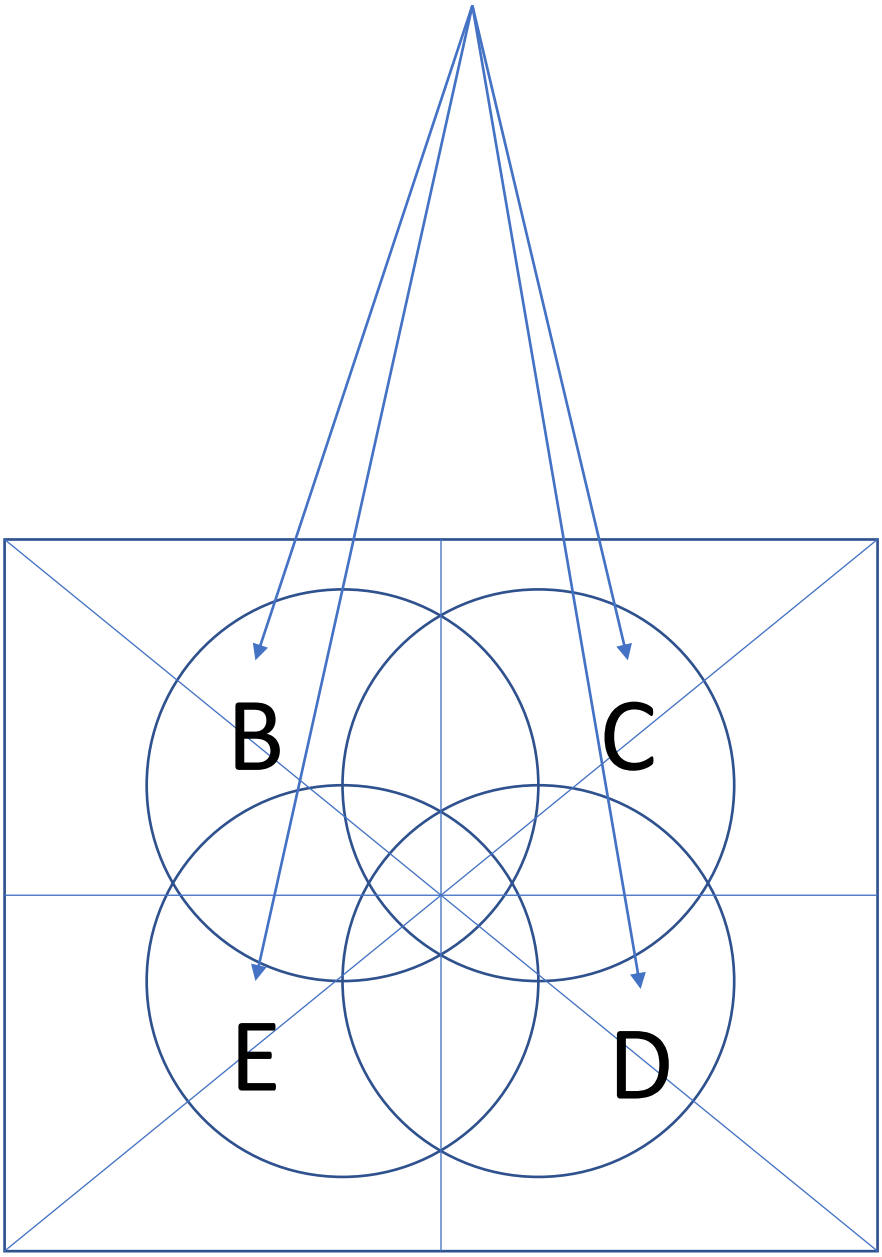
A

Statement



A

Statement



Modernization of Euler's Logical Diagram

Fedorchenko Mikhail Valerievich

Abstract:
This document presents a modern reinterpretation of Euler's diagram by integrating geometric symmetry and logical formalism

1. Classical Euler Logic Overview
Euler diagrams are traditionally used to represent logical relations between sets. For two sets A and B:
- Region only in A: $A \wedge \neg B$
- Region only in B: $B \wedge \neg A$
- Region in both A and B (intersection): $A \wedge B$
- Outside both: $\neg A \wedge \neg B$

2. Modernized Diagram Structure
The modernized version introduces:
- Two overlapping circles representing sets A and B
- A central zone of intersection (C)
- A rectangular frame enclosing the system (space S)
- A vertical and horizontal axis of symmetry

3. Formal Logical Definitions
Let:
- A: Left circle
- B: Right circle
- C: $A \cap B$ — Intersection (Center region)
- XL = $A \wedge \neg B$ — Left exclusive zone
- XR = $B \wedge \neg A$ — Right exclusive zone
- D = $A \oplus B$ — Symmetric difference
- Z = $A \cup B \subseteq S$ — Logical area within space
Symmetry condition:
- $\forall x \in S, \exists x' : x = \text{mirror}(x')$ over vertical axis

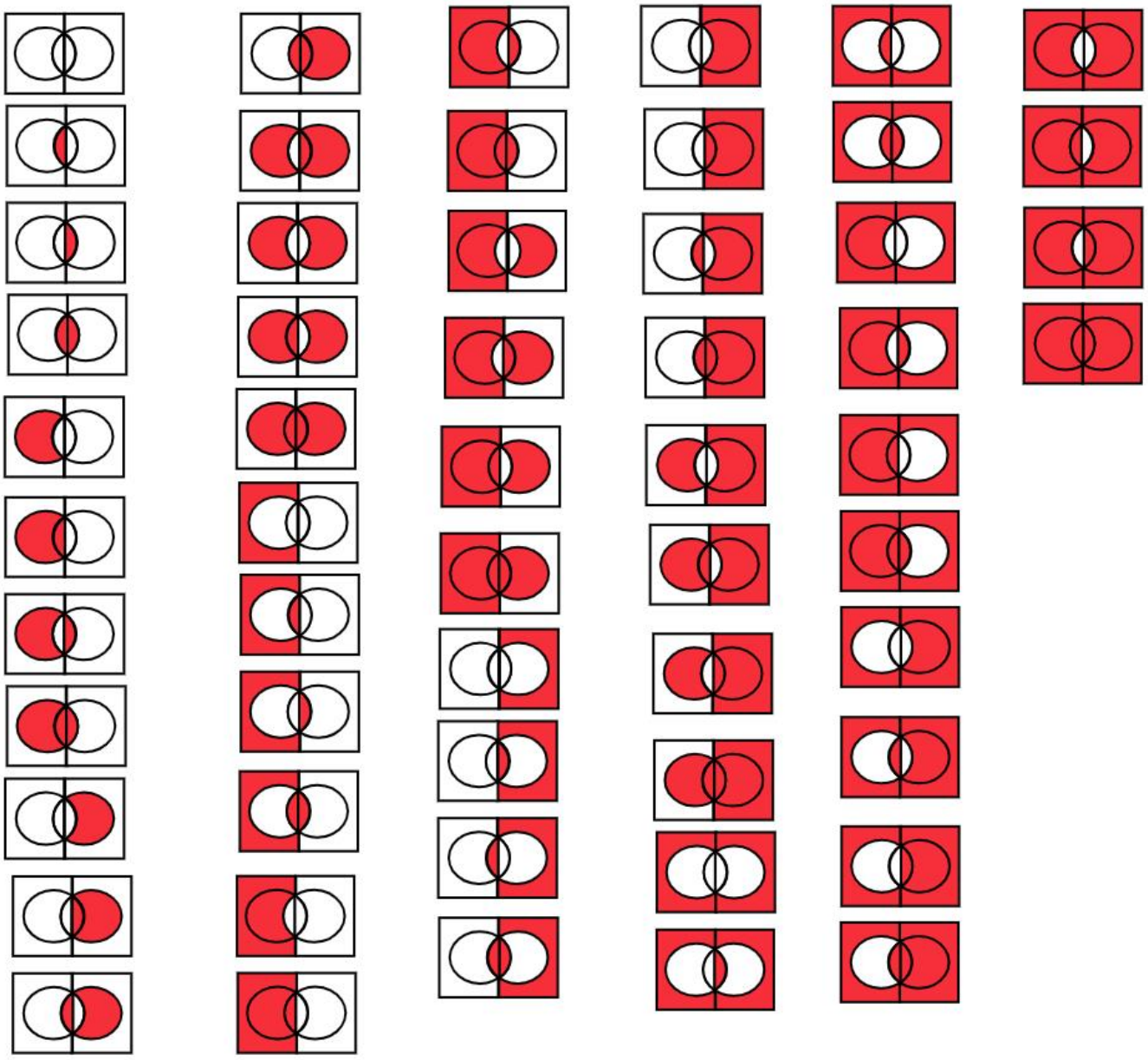
4. Spatial Logical Function (ψ)
We define a function to describe logical values across the space:
$\psi(x, y) =$
1, if $(x, y) \in A$
2, if $(x, y) \in B$
3, if $(x, y) \in C = A \cap B$
0, otherwise
Alternative: $\psi(x, y) = A(x, y) + 2 * B(x, y)$

5. Z-axis Extension: Vertical Logical Separation
We introduce the third dimension:
- $\psi(x, y, z) = \psi(x, y) * f(z)$, $f(z) = 1$ if $z > 0$, -1 if $z < 0$
- Logical inversion: $\psi(x, y, -z) = -\psi(x, y, z)$
- Enables modeling dual-layered systems or quantum-state inversions

6. Applications and Extensions
- Advanced symbolic logic education
- Semantic networks visualization
- Quantum logic topology modeling
- System design with spatial-semantic constraints
- Z-axis extension for multi-level logic modeling

Conclusion:
The modernization of Euler’s diagram combines logical and geometric clarity. Introducing axes, spatial logic, and functional ma

Keywords: Euler diagram, modern logic, spatial logic, intersection, symmetry, formal system, z-axis logic

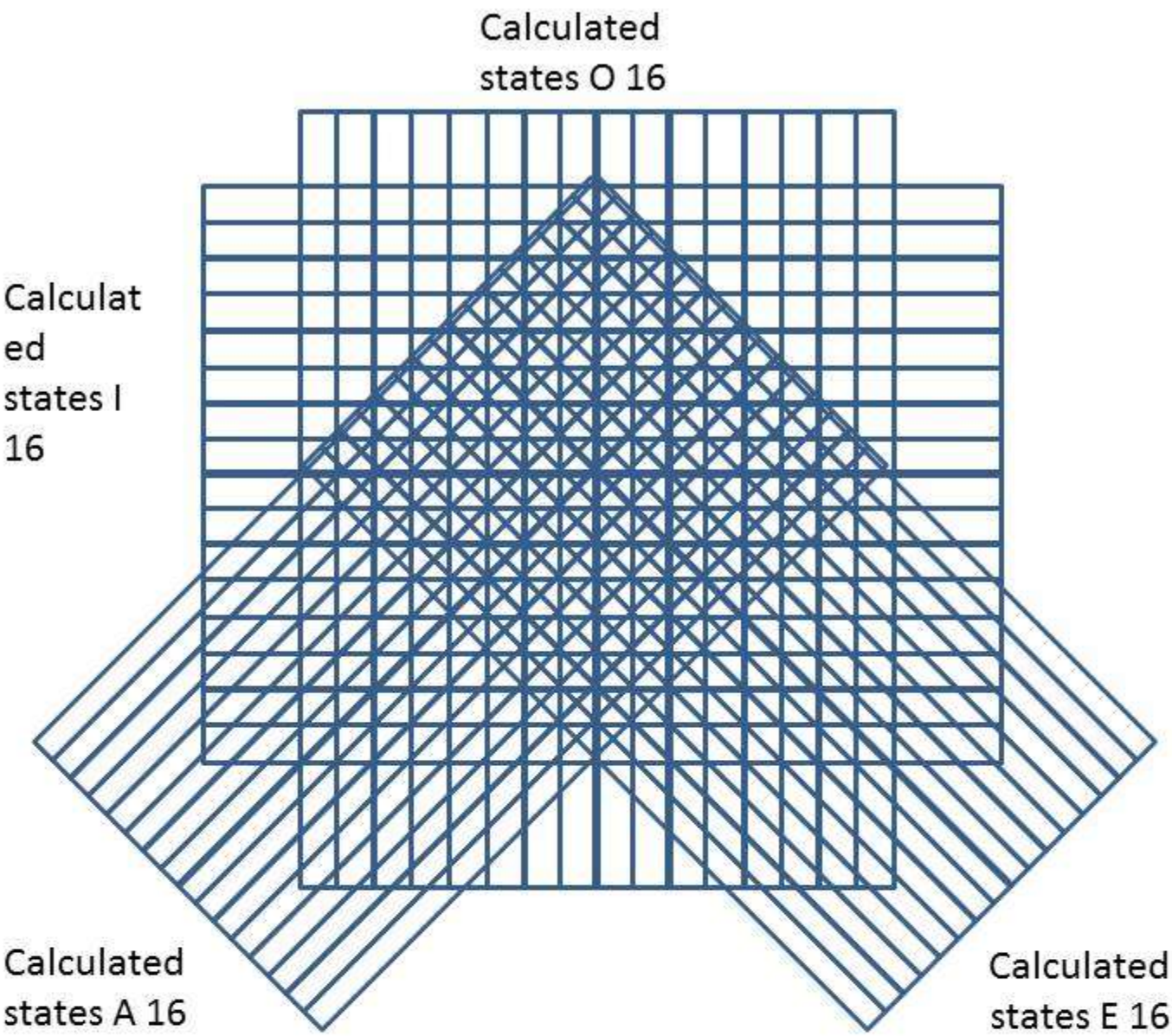


Федорченко Михаил Валерьевич

Second order logic.

Ни в коем случае не сразу в ИИ, сначала человек или параллельно человек и ИИ.

In no case immediately in AI, first a person or in parallel a person and AI.



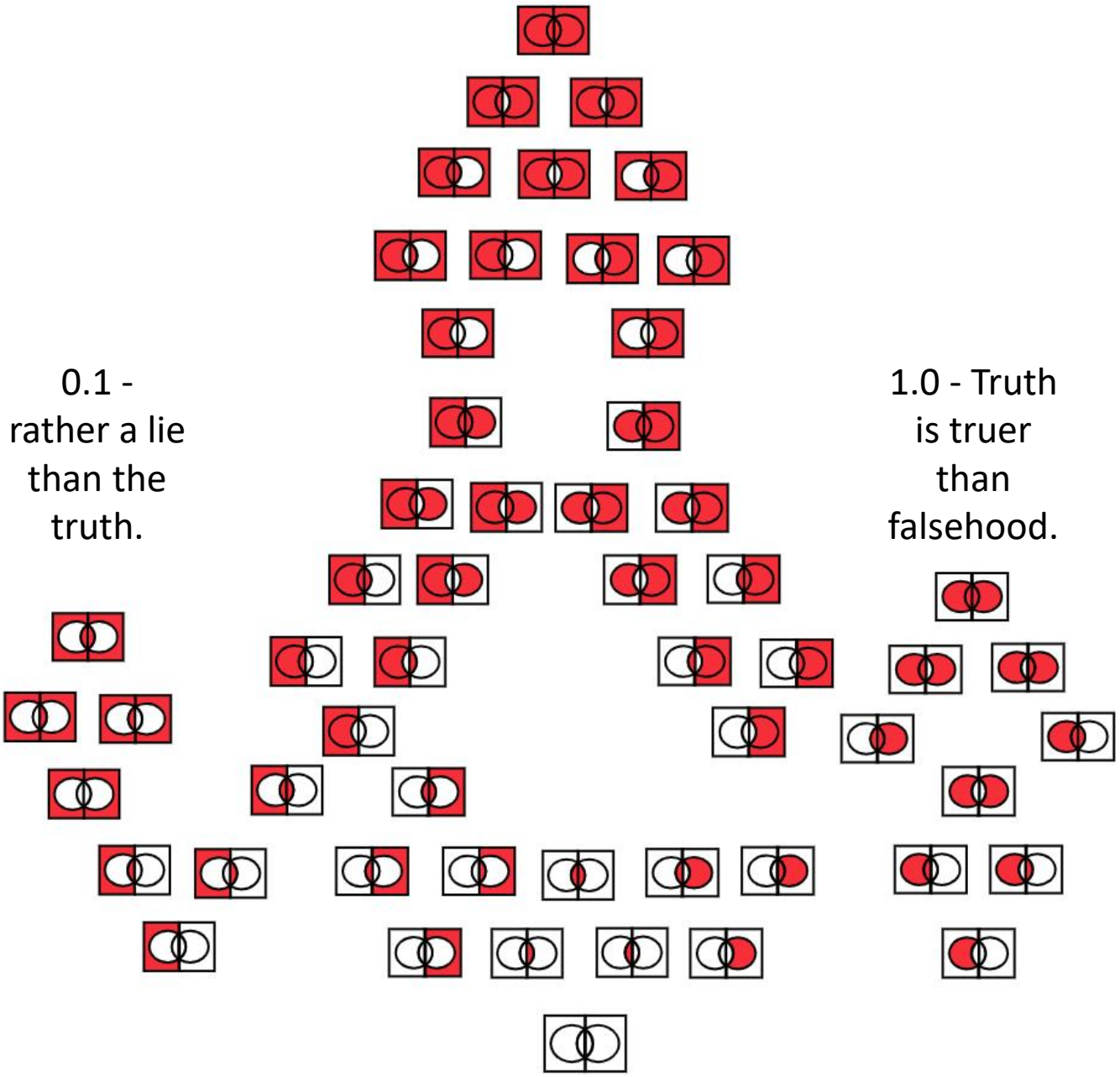
Fedorchenko Mikhail Valerevich

16 logic states = 18 bits = IQ 150+
64 logic states = 72 bits = IQ 600+
256 logic states = 288 bits = IQ 2400+
With 100% Y chromosome = 100% IQ = 5000.

1.1 - True

0.1 - rather a lie than the truth.

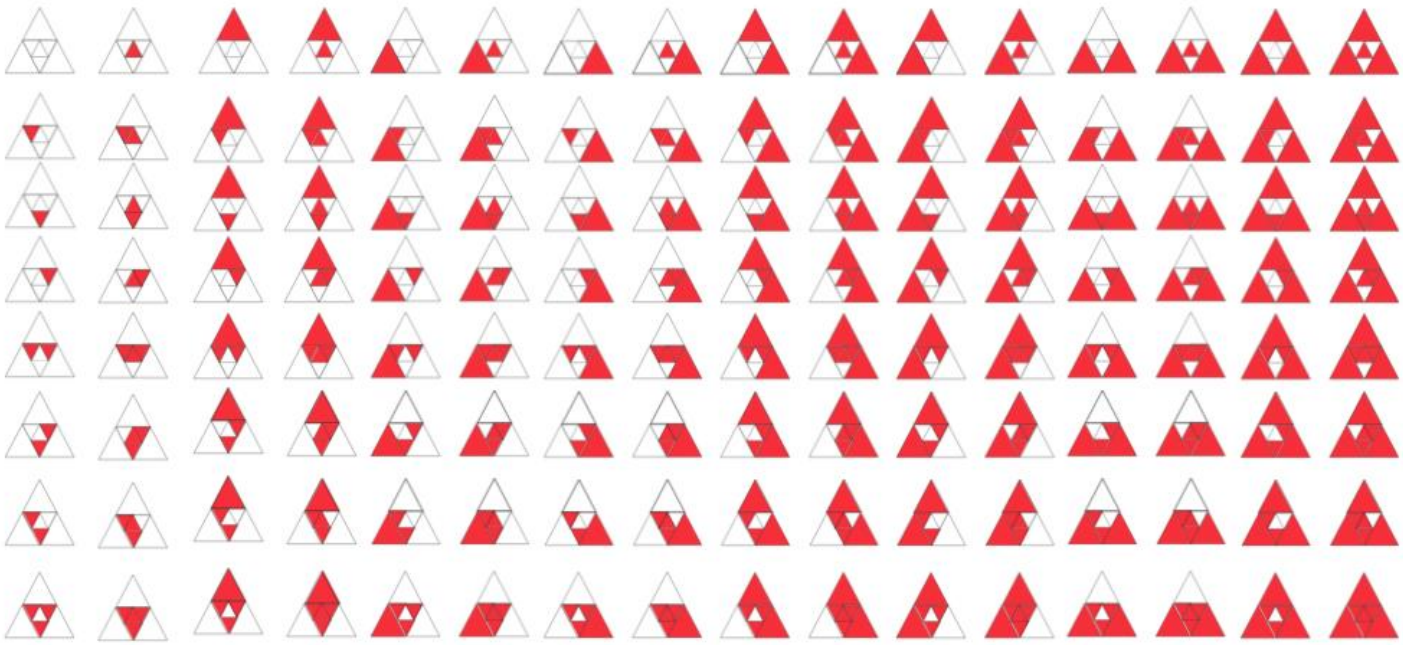
1.0 - Truth is truer than falsehood.



Федорченко Михаил Валерьевич

0.0 - False

Syllogistics intellectus cogniutum
128



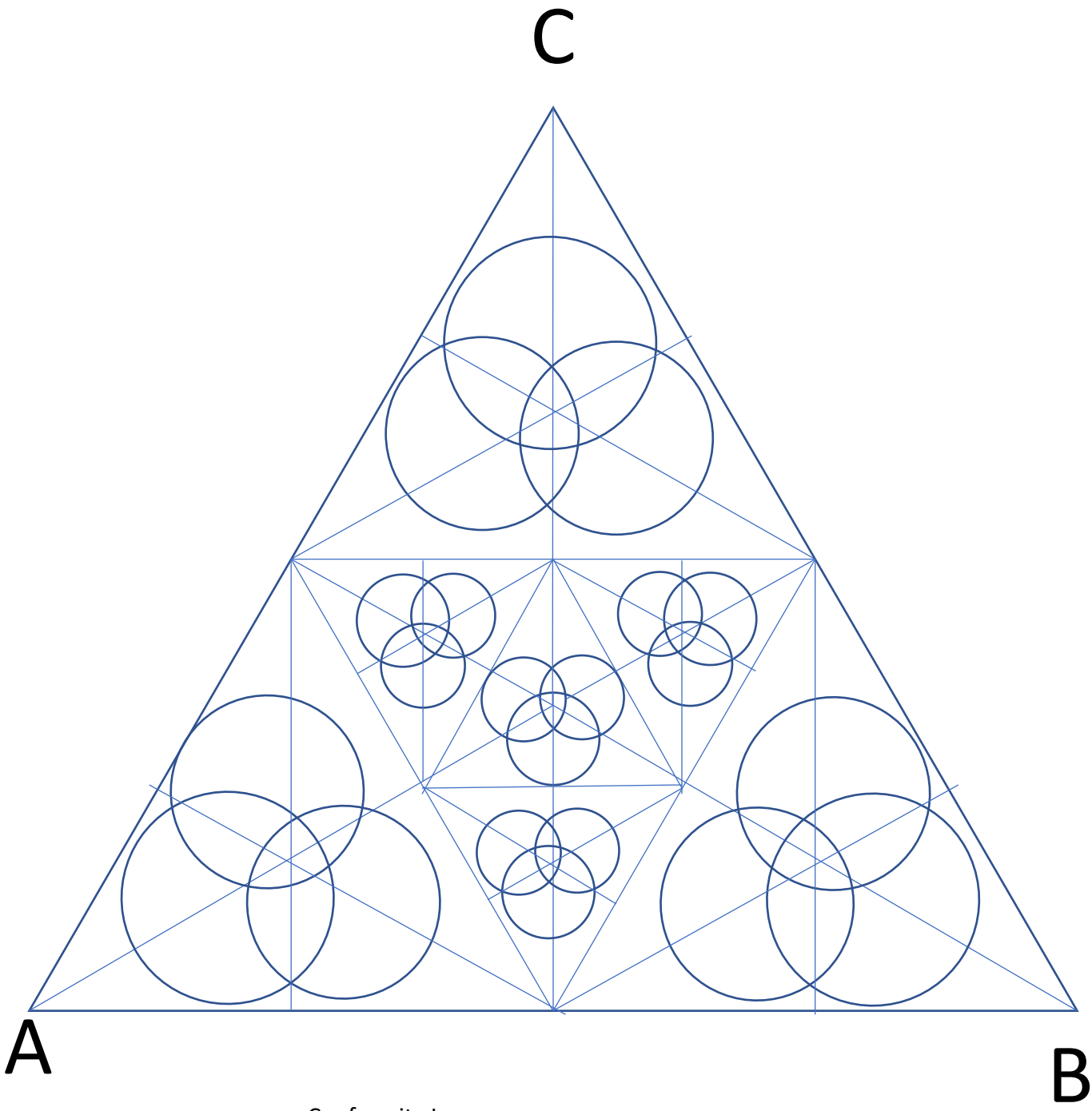
Федорченко Михаил Валерьевич.

Это в первую очередь для Разума Человека, а не машины.

This is primarily for the Human Mind, not the machine.

Закон логики: Никогда не используйте то, что спорно и вызывает разные интерпритации и выводы, там, где нет права на ошибку.

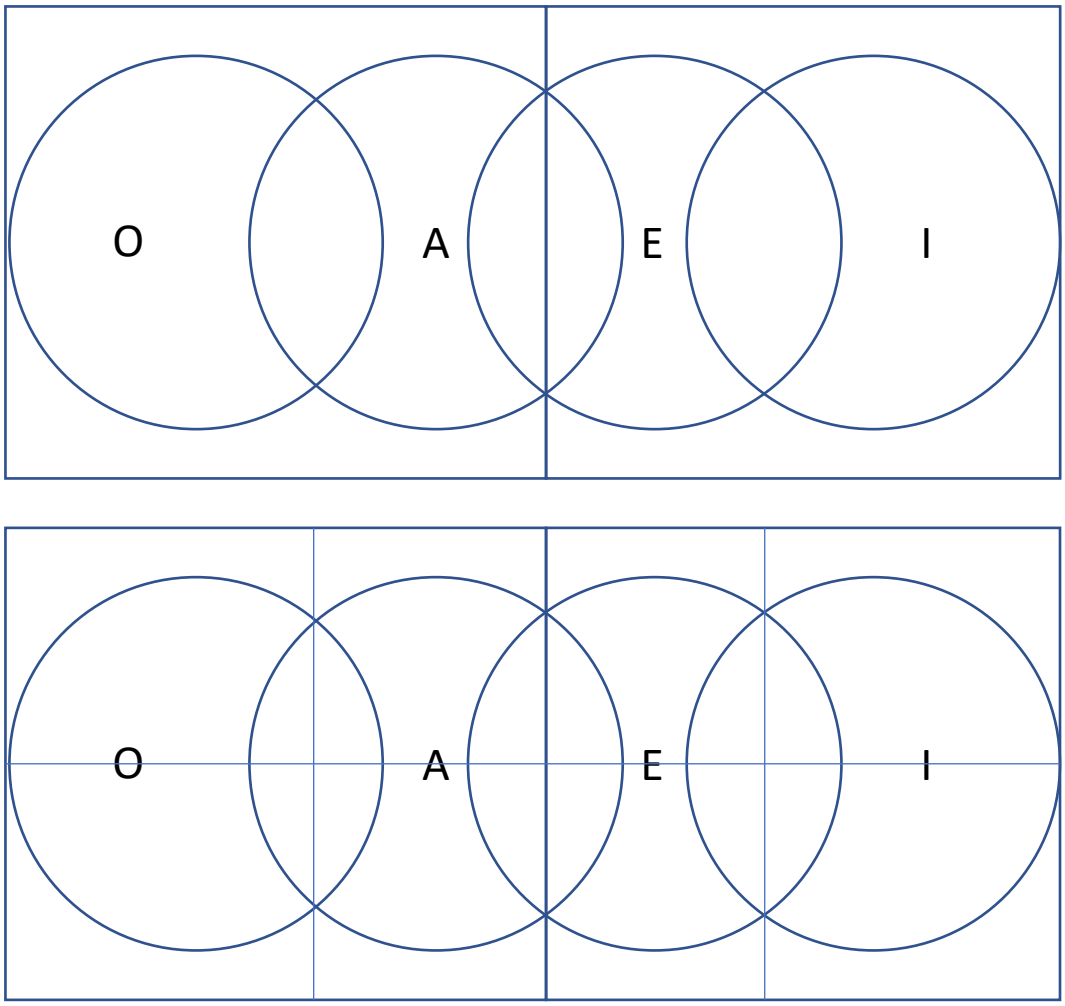
Law of Logic: Never use something that is controversial and gives rise to different interpretations and conclusions, where there is no room for error.



Conformity Law :
A = A , A not B
A1=1 A2=2 => A1 != A2
A=1 B=1 1=1 A not B
Save of Life and World.

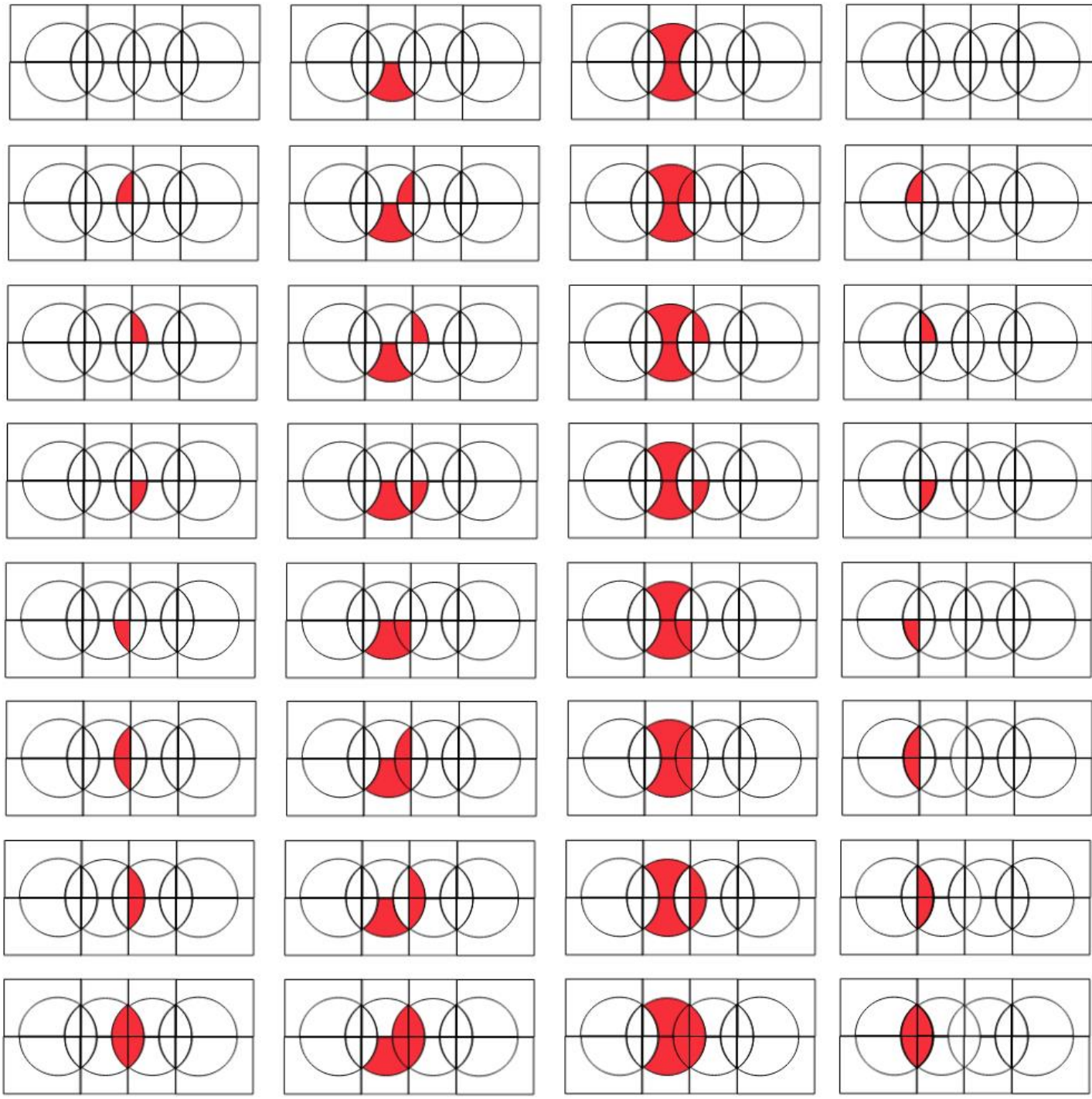
If the Law of Comformityis broken or distorted,
and for AI, all are Doomed.

Fedorchenko Mikhail Valerievich



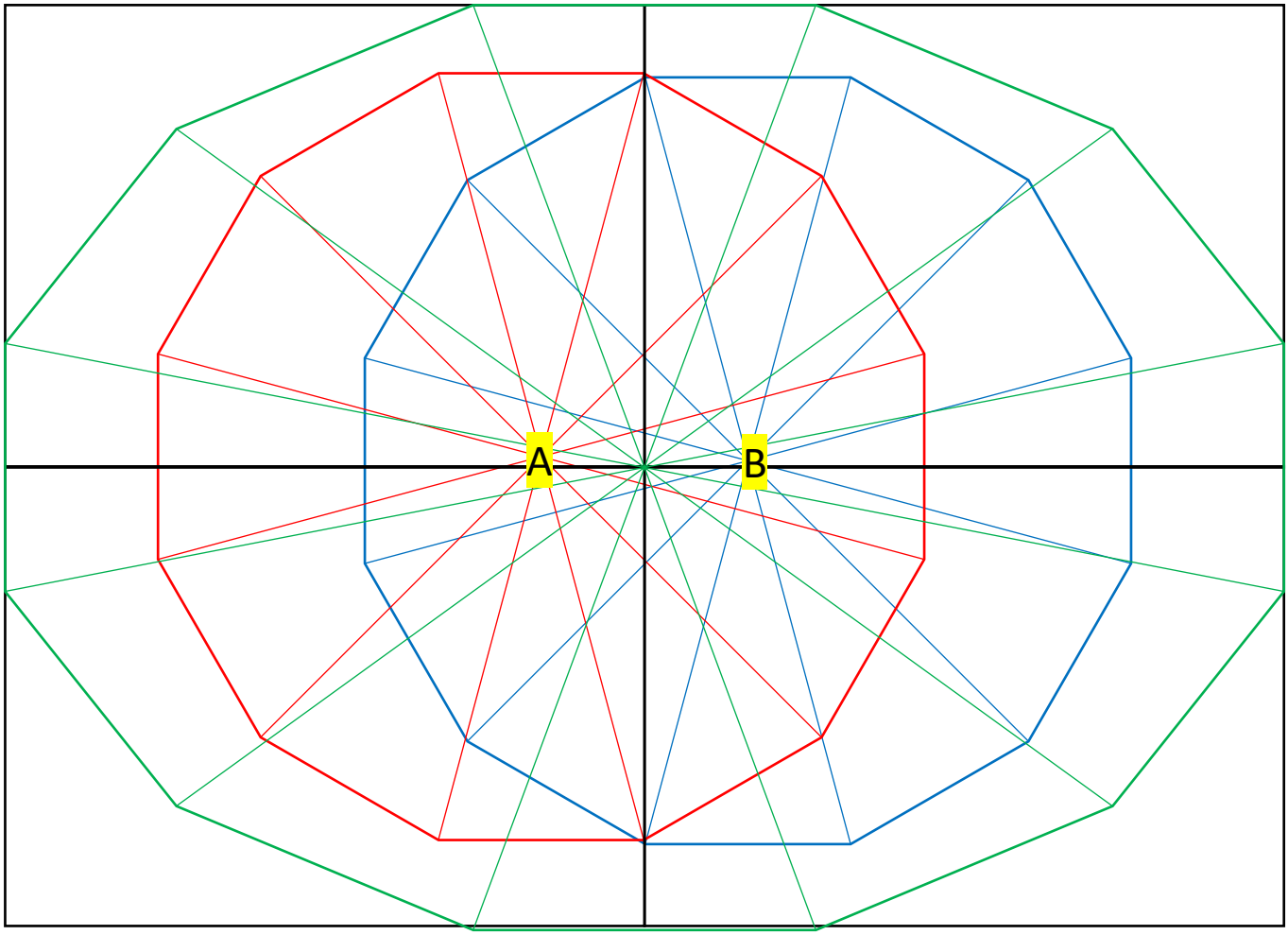
Euler–Fedorchenko (Heron) diagram.

Fedorchenko Mikhail Valerievich

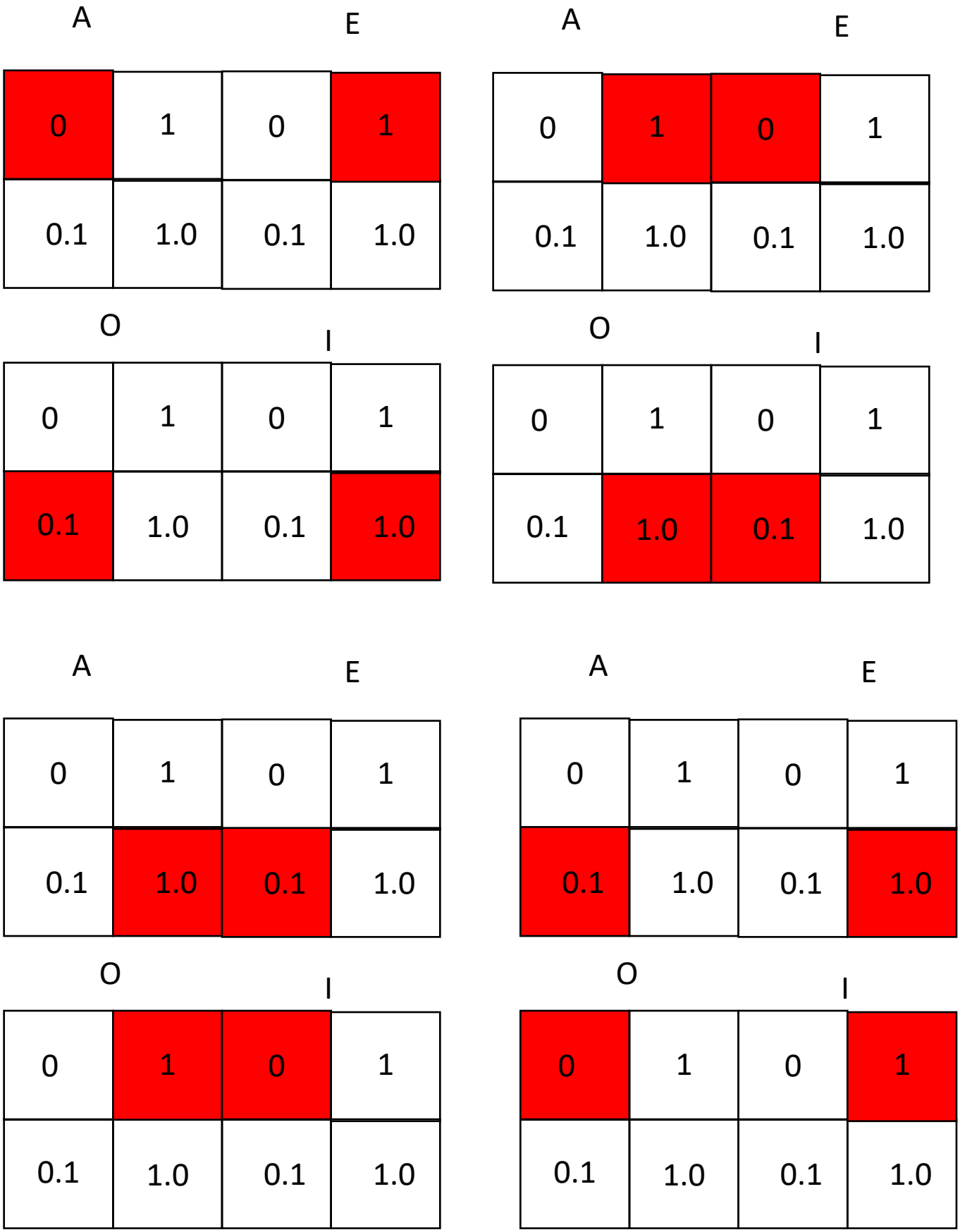


Fedorchenko Mikhail Valerevich

Super Quality Logic



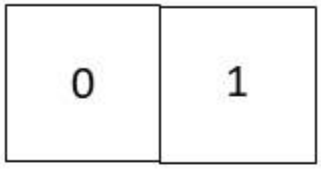
Fedorchenko Mikhail Valerevich



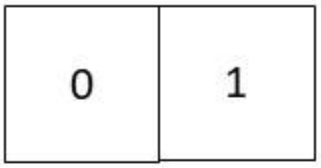
Procedural solution of the logical square, 4 steps, based on modernized logic.

Fedorchenko Mikhail Valerevich

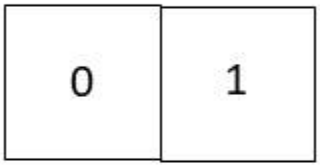
A



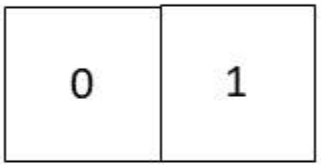
E



O



I

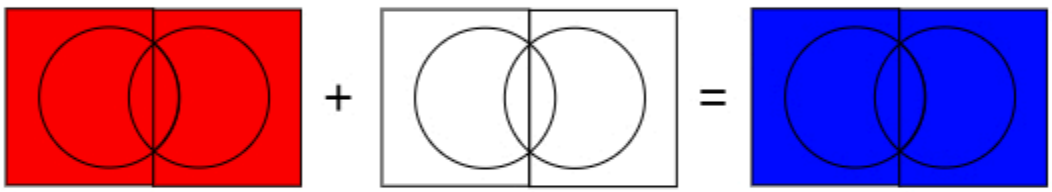


Now try to solve this problem with only 1 and 0. The rules of classical logic have been rewritten and distorted in a biased manner.

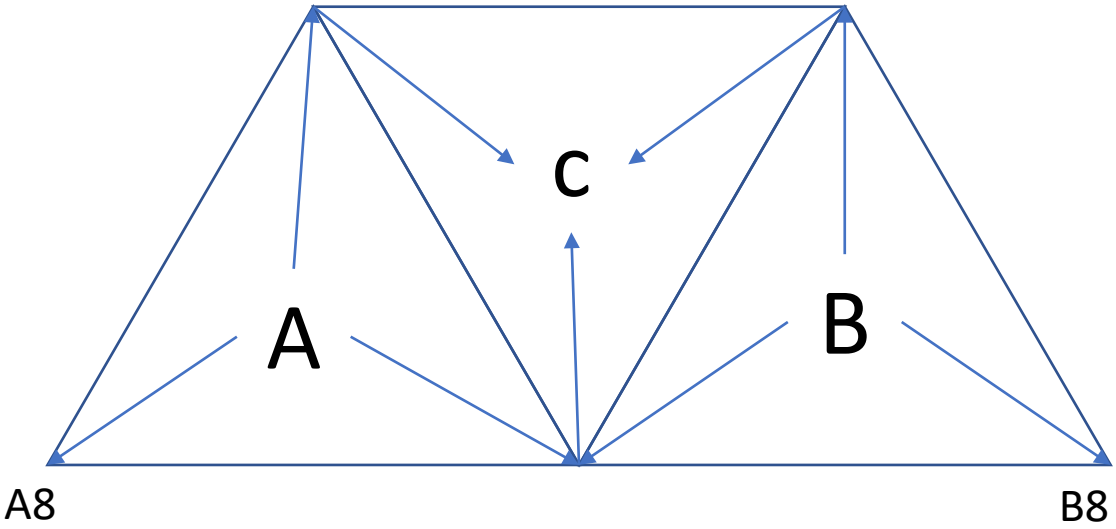
This is an example of a monstrous error embedded in logic, the purpose of which is zombification, radicalization, extremism, as well as the production of tyranny and totalitarian regimes, totalitarian and limited thinking and consciousness.

Fedorchenko Mikhail Valerevich

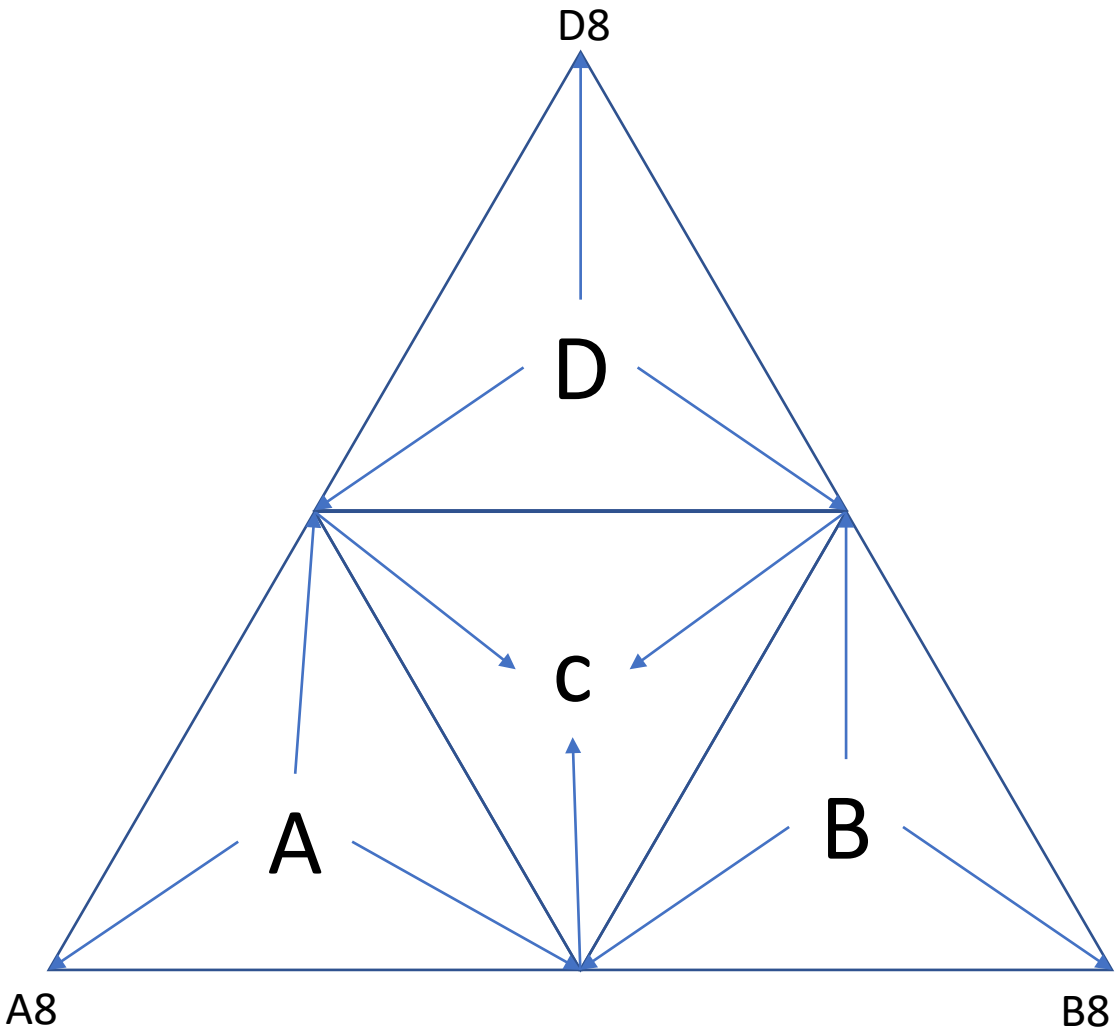
Classic Logic is not original.



Fedorchenko Mikhail Valerevich

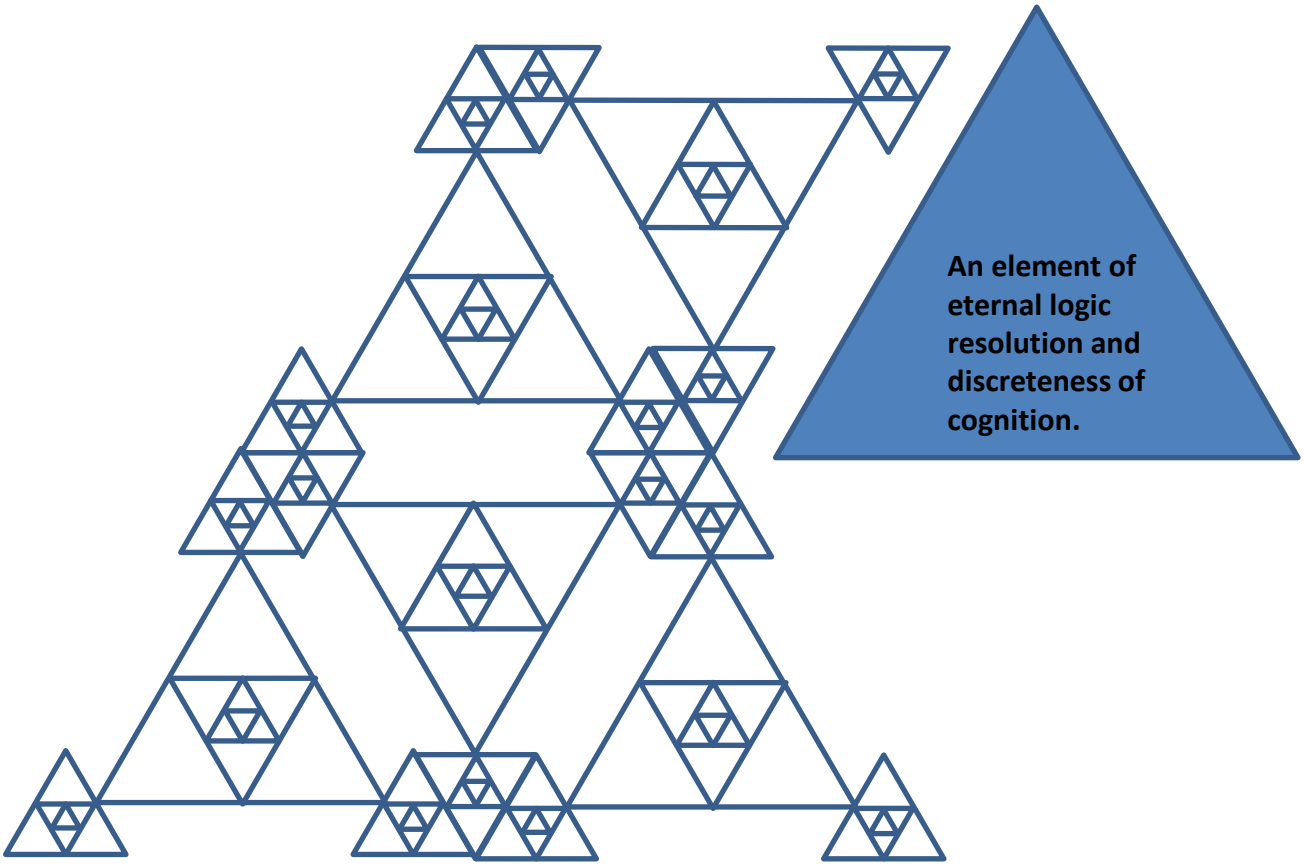


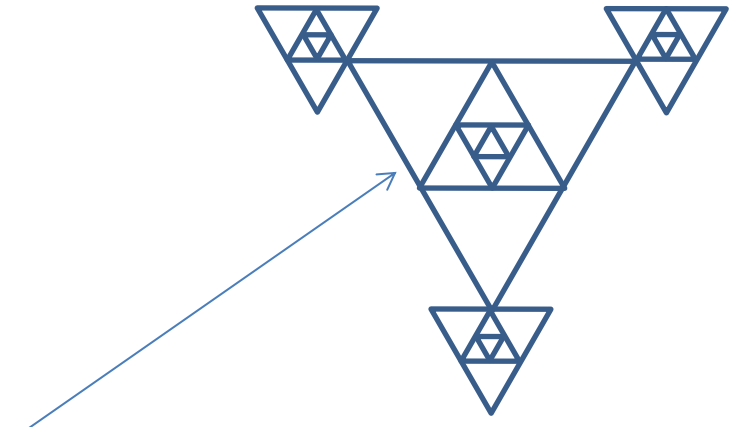
A8+B8 ->C1 validate ->C



A8+B8+D8 ->C1 validate ->C

Logic of the Great, Logic of the Wise
Fedorchenko Mikhail Valerevich



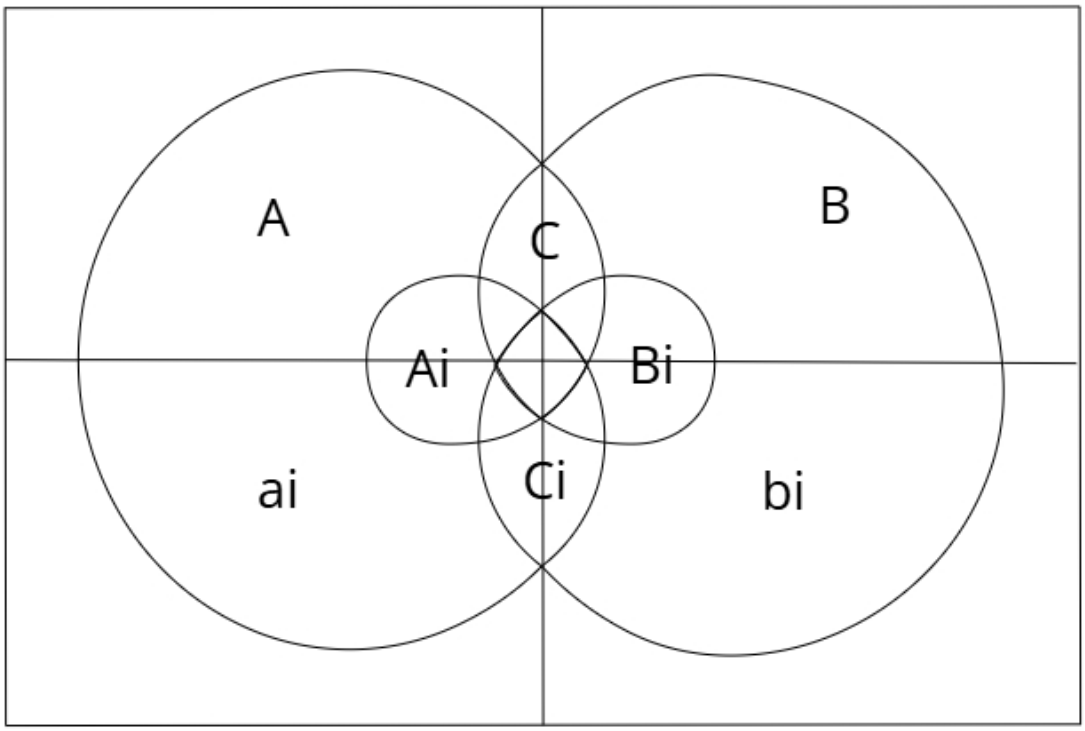


Unit Matrix of Hegel + Syllogistic intellectus cognitum

Europe brought in Marx, Nietzsche, Heidegger, who turned Europe into a hungry and impoverished space for the lumpines, something that Hegel tried to prevent.

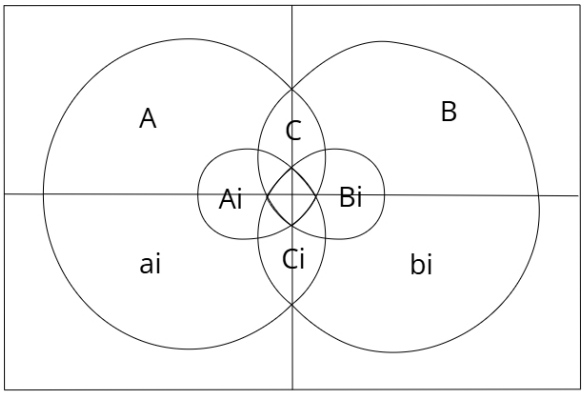
Федорченко Михаил Валерьевич

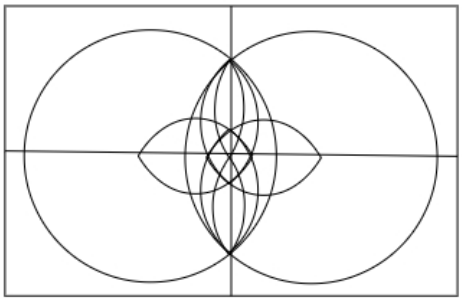
Formula of Pythagorean + Great logic:
$(\bigcup i(Ai \cap Bi)) \longrightarrow S$ where $Ai \cap Bi$ but $i(A) \bigcup i(B)$
where:
Ai and Bi are different points of view, opposing concepts or states of understanding.
$\cap$ is the intersection of points of view or conflict of ideas, their clash.
$\bigcup$ is the unification of different intersections and points of view, the creation of the set of all possible intersections.
$\longrightarrow$ is the transition from one state to another, the evolution of thinking.
S is synthesis, higher understanding, harmony, the result of the interaction of different points of view and transitions between them.
Explanation of the formula:
Intersection of points of view ($Ai \cap Bi$): Each pair of opposing ideas or views interacts in the process of collision and intersection, which serves as the beginning of growth and revelation of truth. Contradictions do not destroy each other, but give rise to new understanding.
Unification ($\bigcup$): All intersections of viewpoints are brought together into a larger picture, where their interactions lead to synthesis and understanding. This means that the logic of the Great Ones includes the ability to connect and consider different aspects in one holistic perception.
Transitions ($\longrightarrow$): Logic is not linear, and each new understanding becomes a consequence of the previous one, but in the process of transition, different levels of thinking intertwine, which allows for new discoveries and growth.
Synthesis and Harmony (S): This is the final goal where the whole process leads - the transition to a higher state of understanding, to harmony, in which opposites, transitions and intersections become part of a single system.

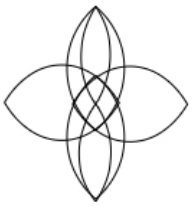


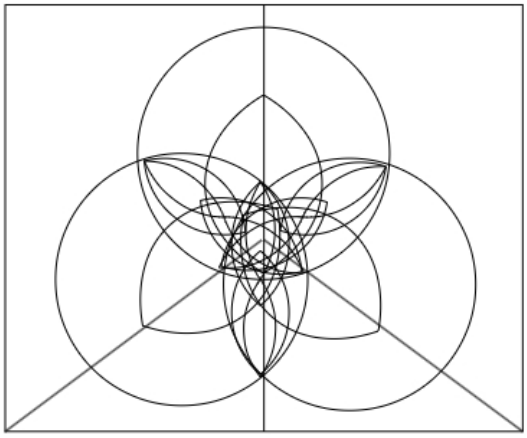
(∪i(Ai∩Bi))⟶S where Ai ∩ Bi but i(A)∪ i(B)

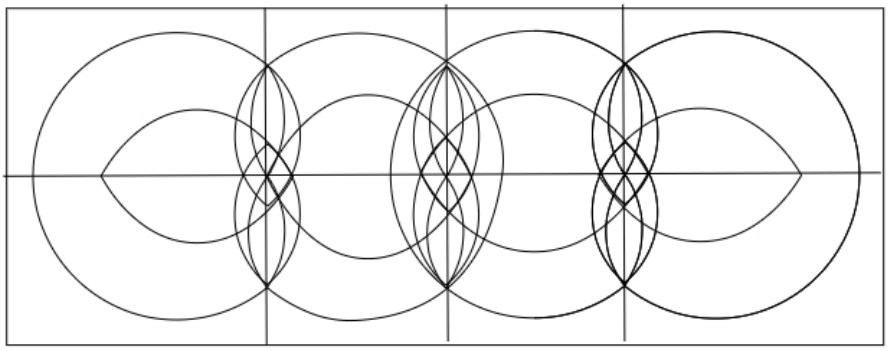
Pythagorean Logic + Logic of the Greats ->
Diagram of Heron (Fedorchenko M.V.)







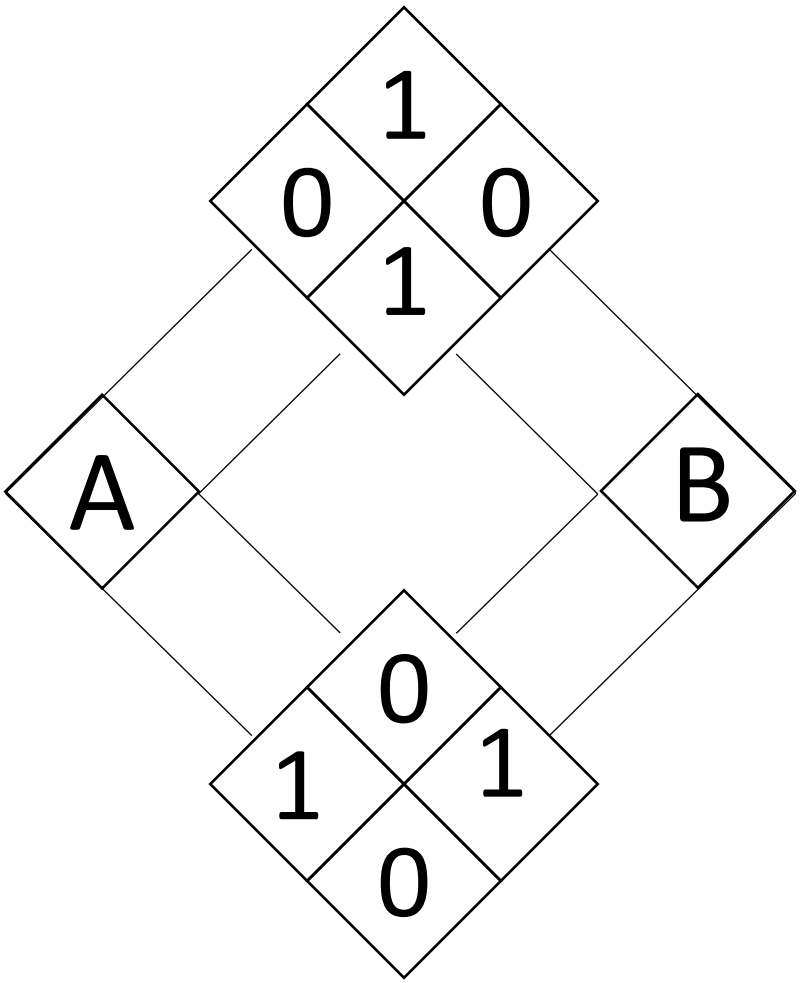




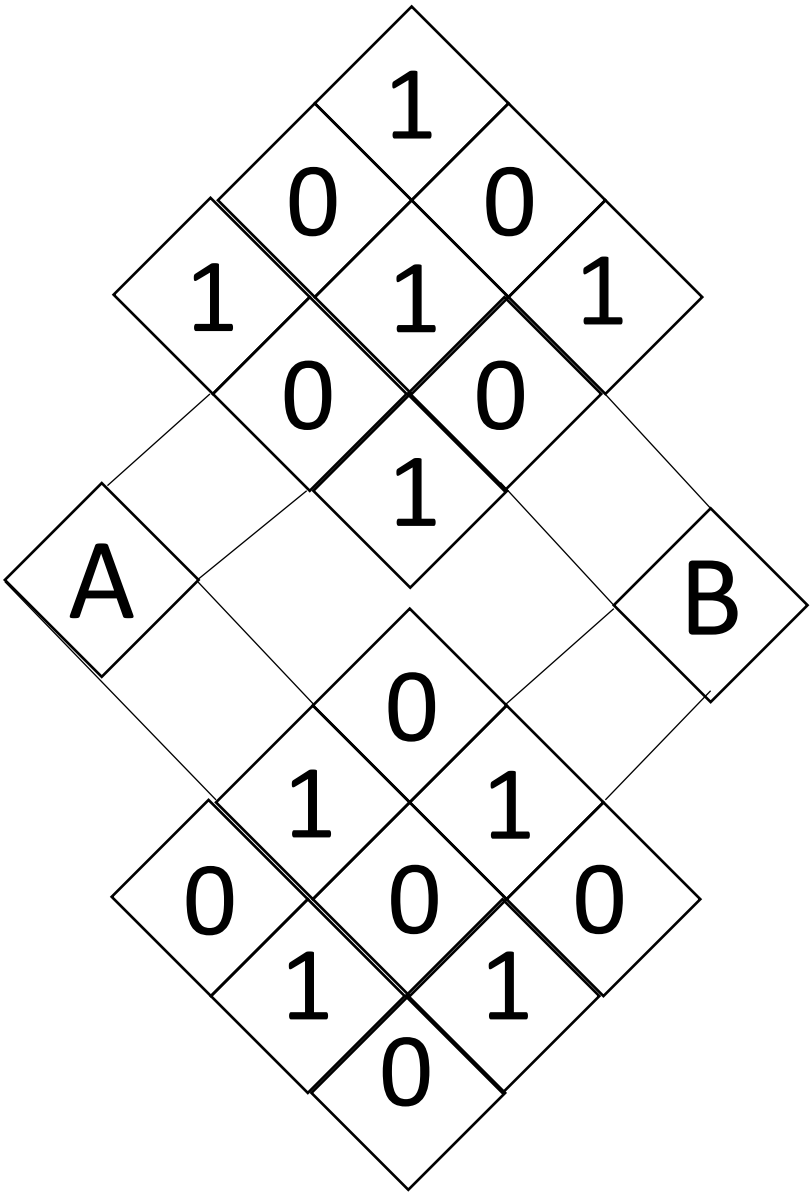
Pythagorean Logic + Logic of the Greats ->
Diagram of Heron (Fedorchenko M.V.)

Величие страны и народа складывается из Успехов его Граждан.

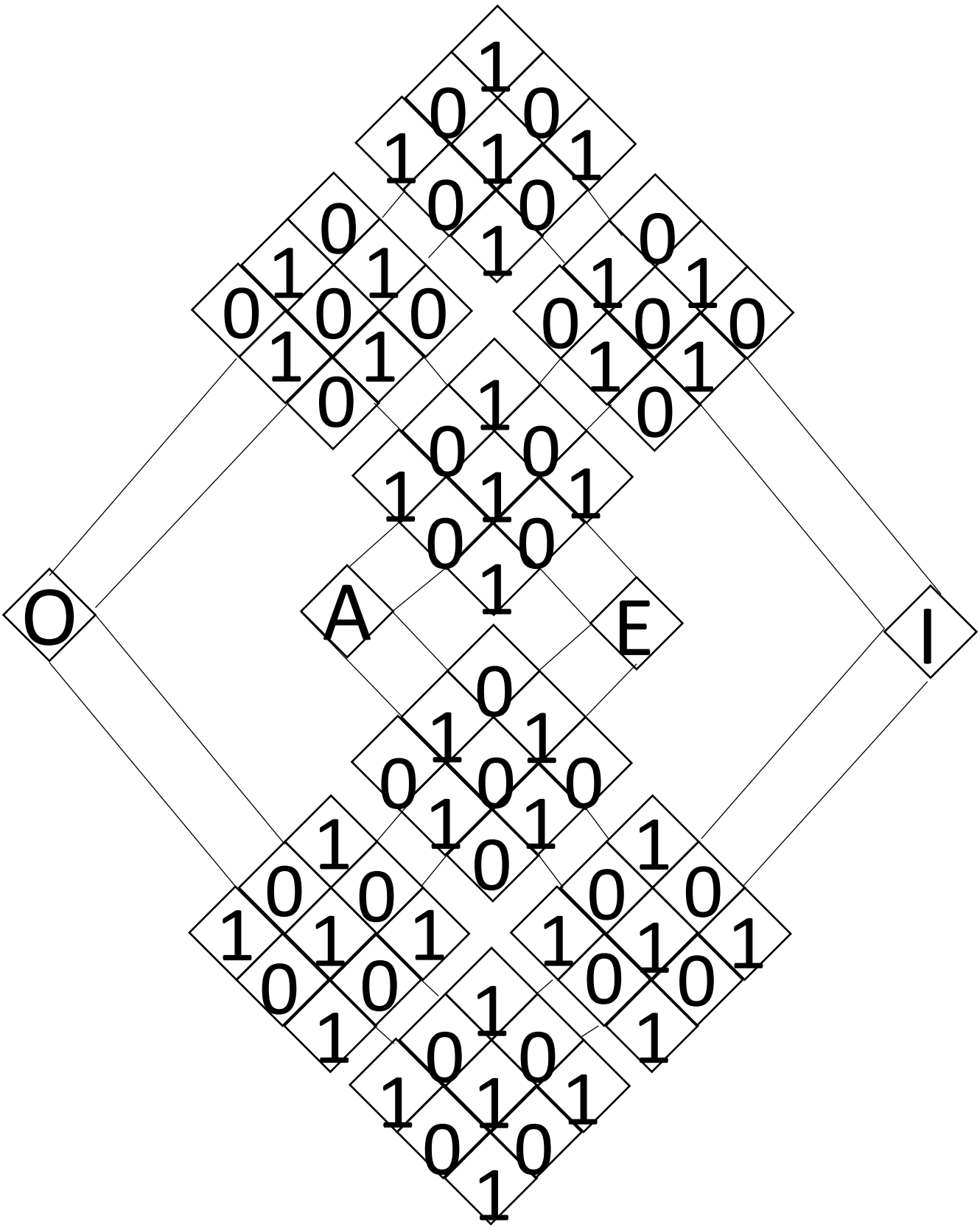
The greatness of a country and its nation is determined by the successes of its citizens.



Pythagorean Logic + Logic of the Greats ->
Diagram of Heron (Fedorchenko M.V.)



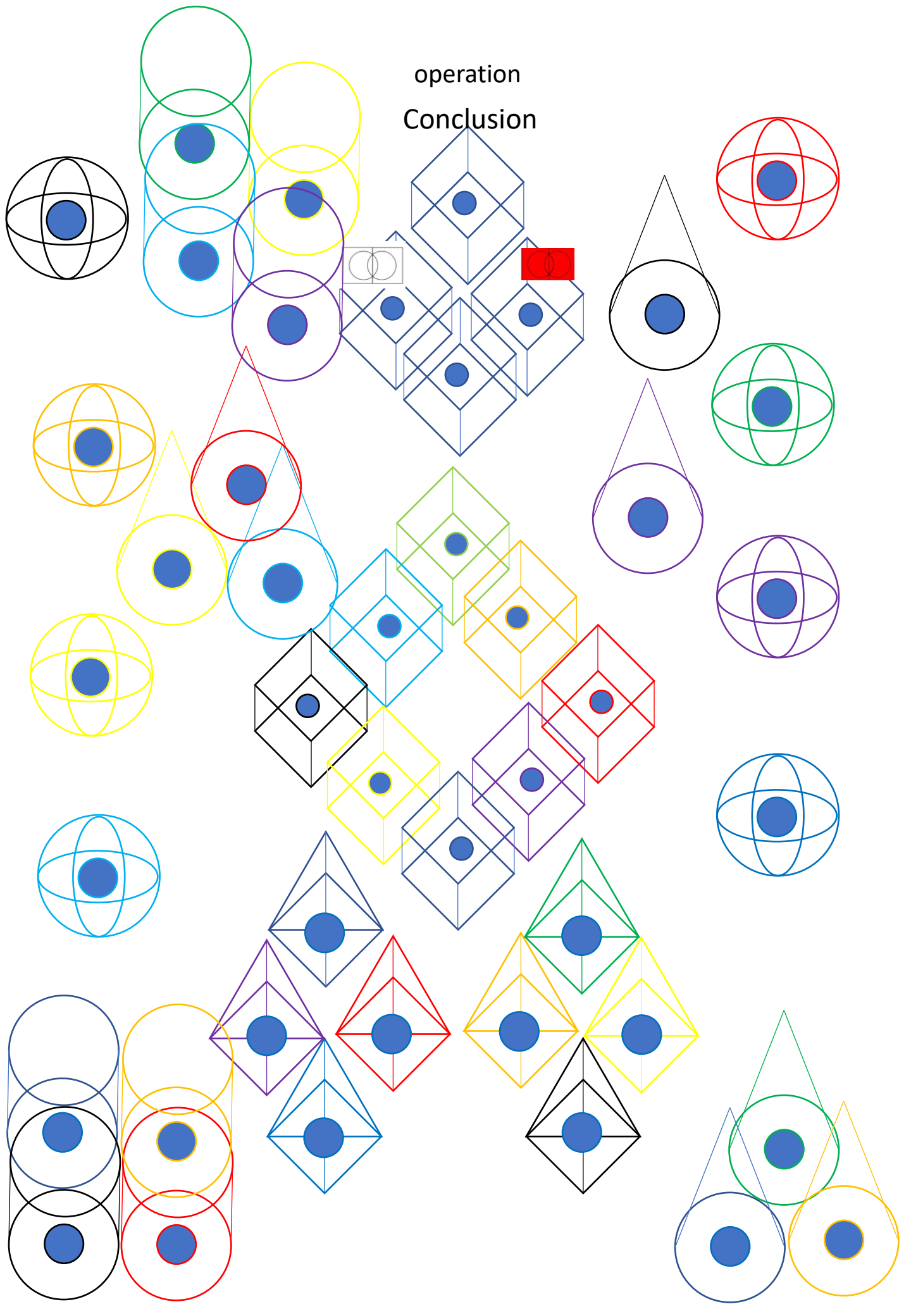
Pythagorean Logic + Logic of the Greats ->
Diagram of Heron (Fedorchenko M.V.)

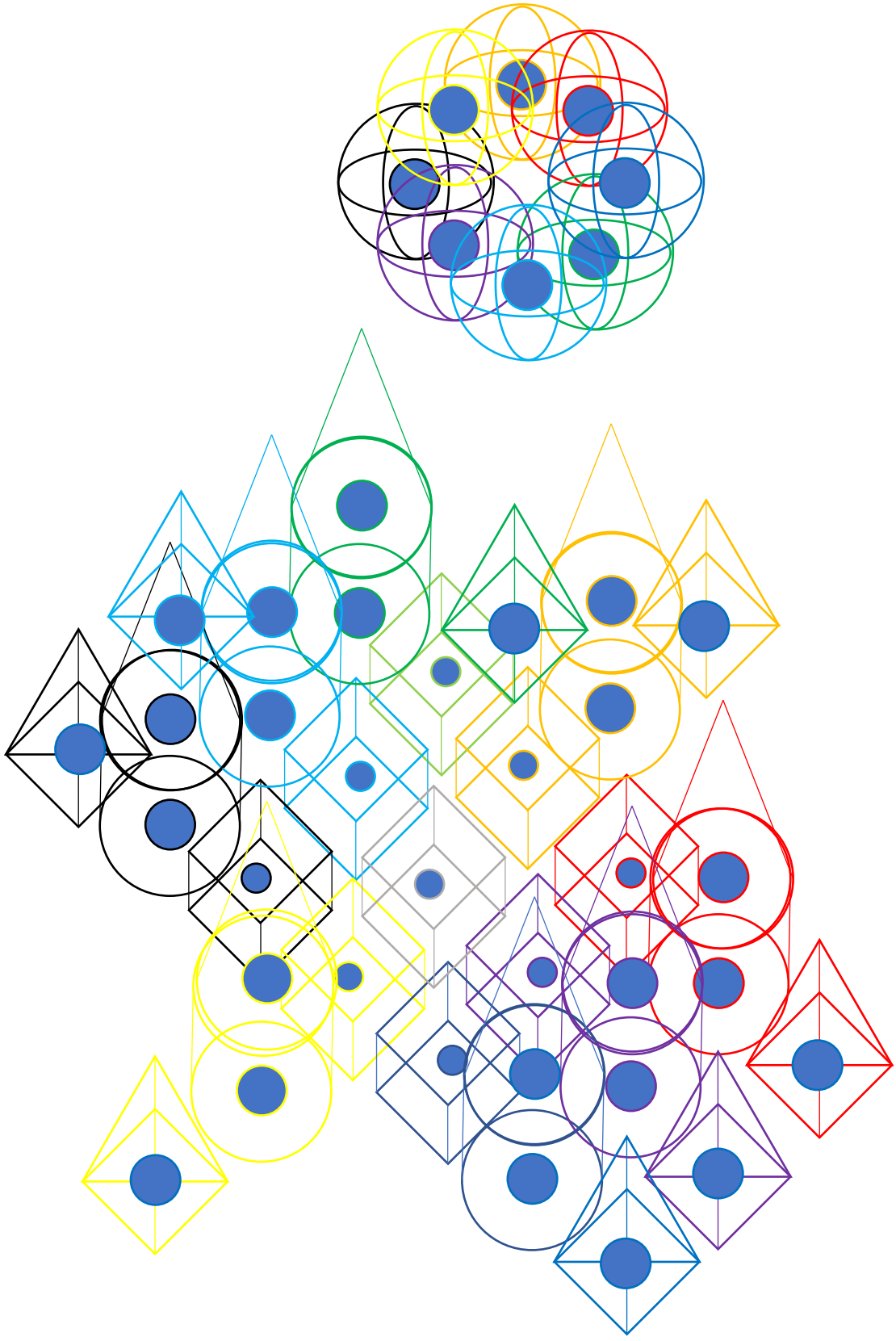


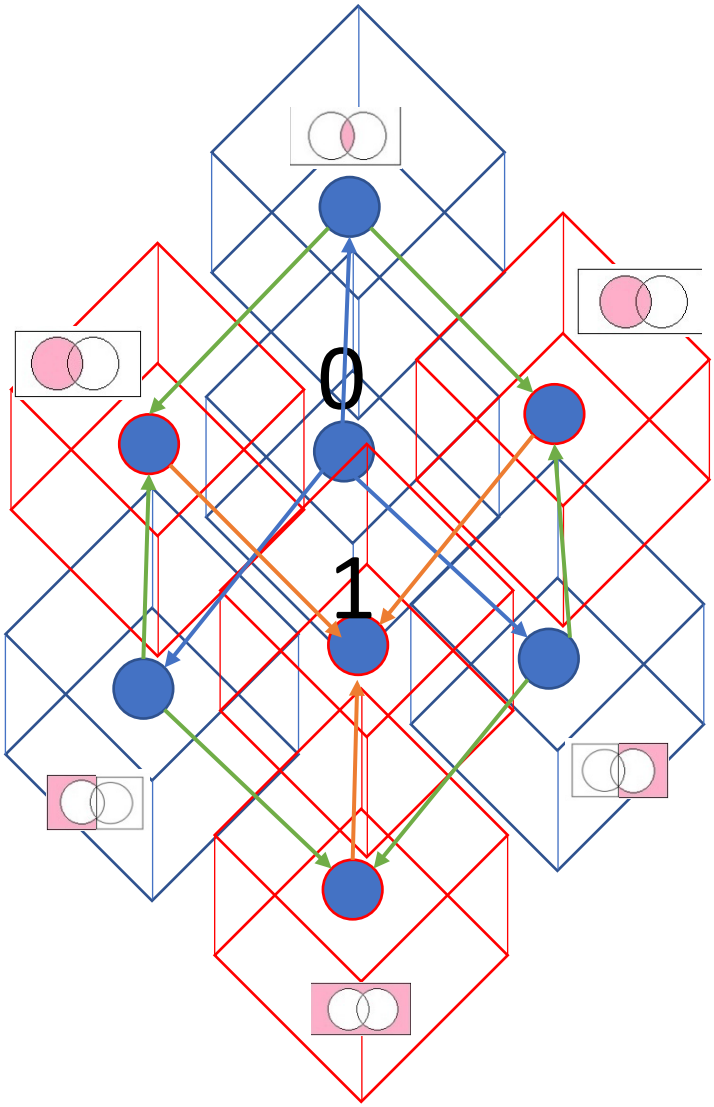
Pythagorean Logic + Logic of the Greats ->
Diagram of Heron (Fedorchenko M.V.)

operation

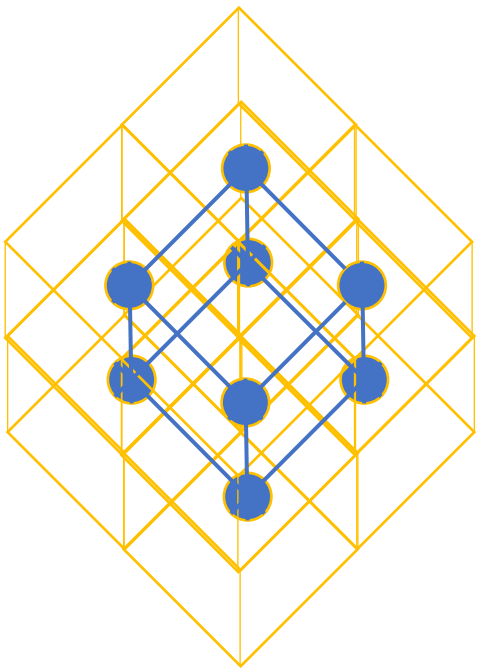
Conclusion

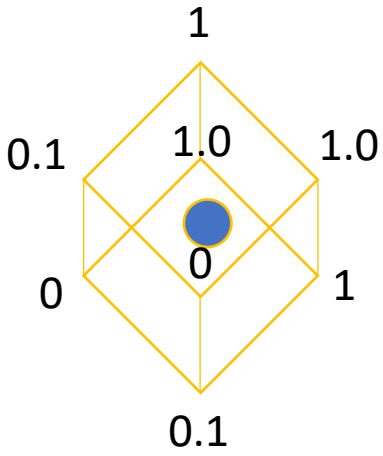






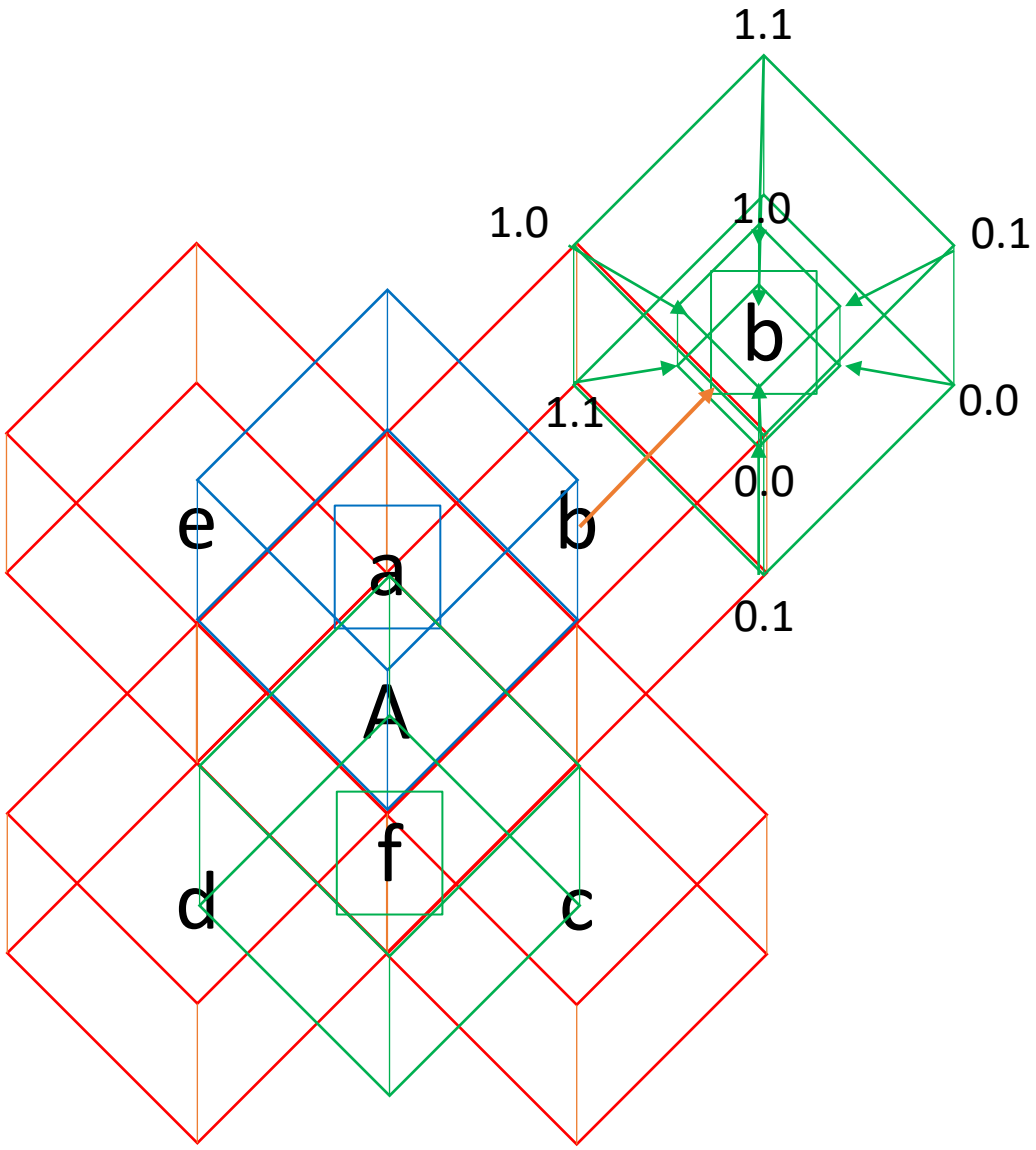
Model of Reasoning
Logic of the Great, Logic of the Wise
Fedorchenko Mikhail Valerevich





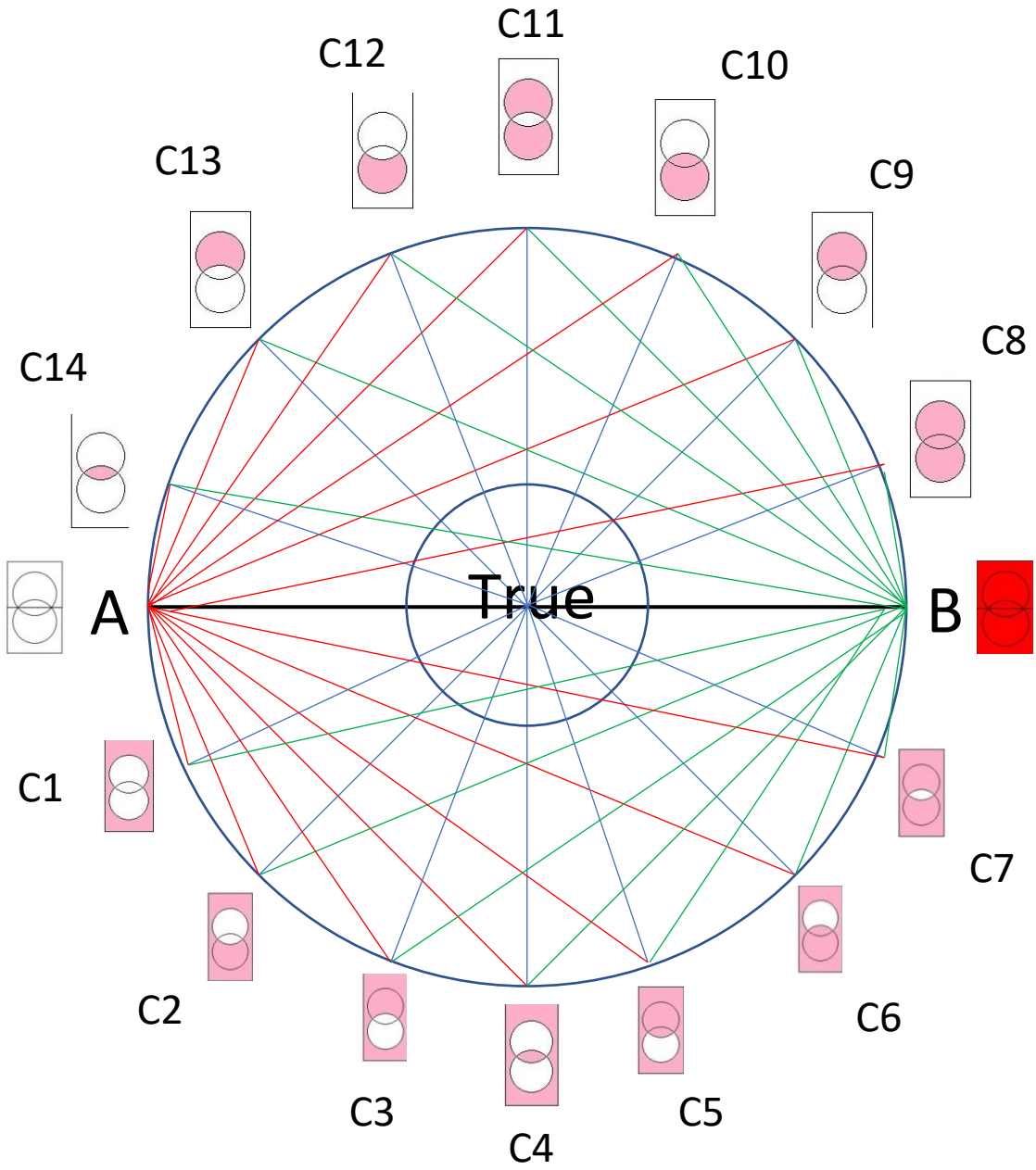
Quantum logic

Fedorchenko Mikhail Valerevich



A-> a,b,c,d,e,f
a = b,c,d,e
f = -b,-c,-d,-e
d= -e,-a,-c,-f
b=e,a,c,f
c=a,b,f,d
c=-a,-b,-f,-d

Logic of the Great, Logic of the Wise
Fedorchenko Mikhail Valerevich

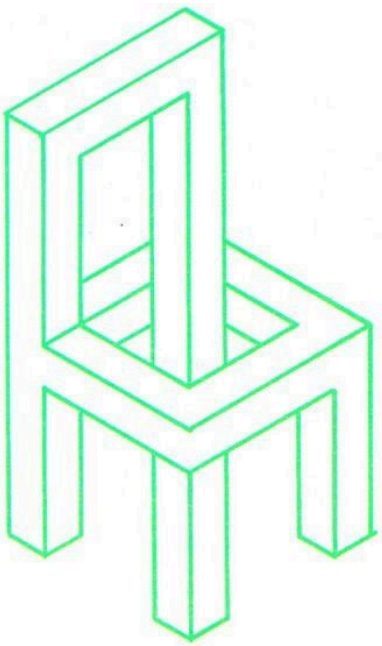


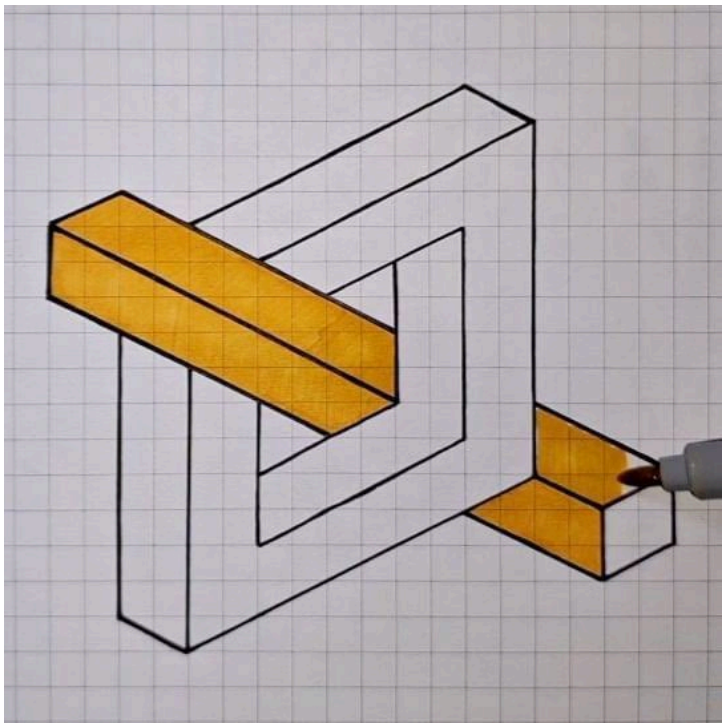
C1+ C8 = 1 ….

A+B = not 1

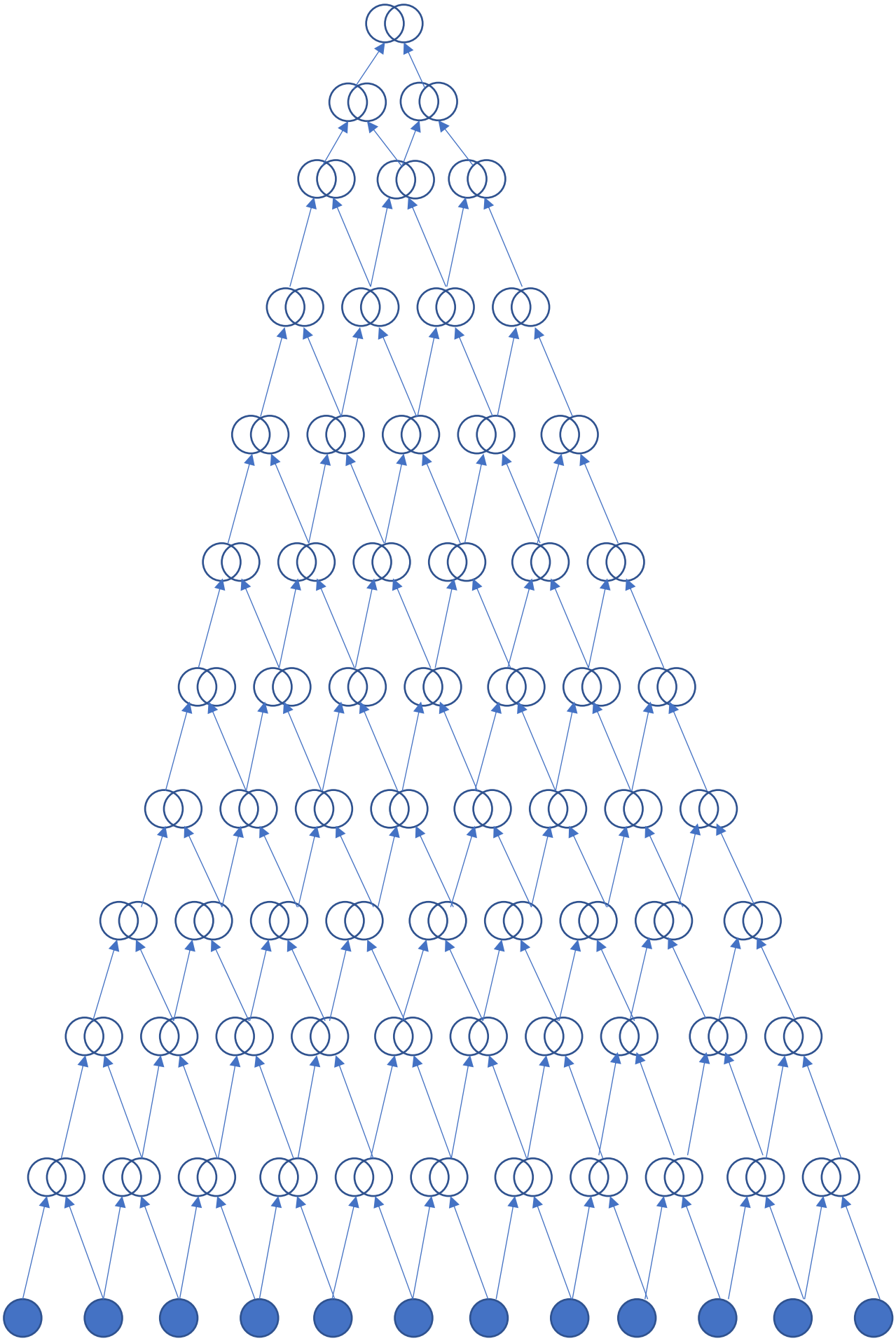
Logic of the Great, Logic of the Wise
Fedorchenko Mikhail Valerevich

Mathematical logic destroys the logical laws of identity and conformation well.
It is possible mathematically, but destructive in understanding Nature and Reality. Mathematics destroys Logic by its intervention.





Decision making system

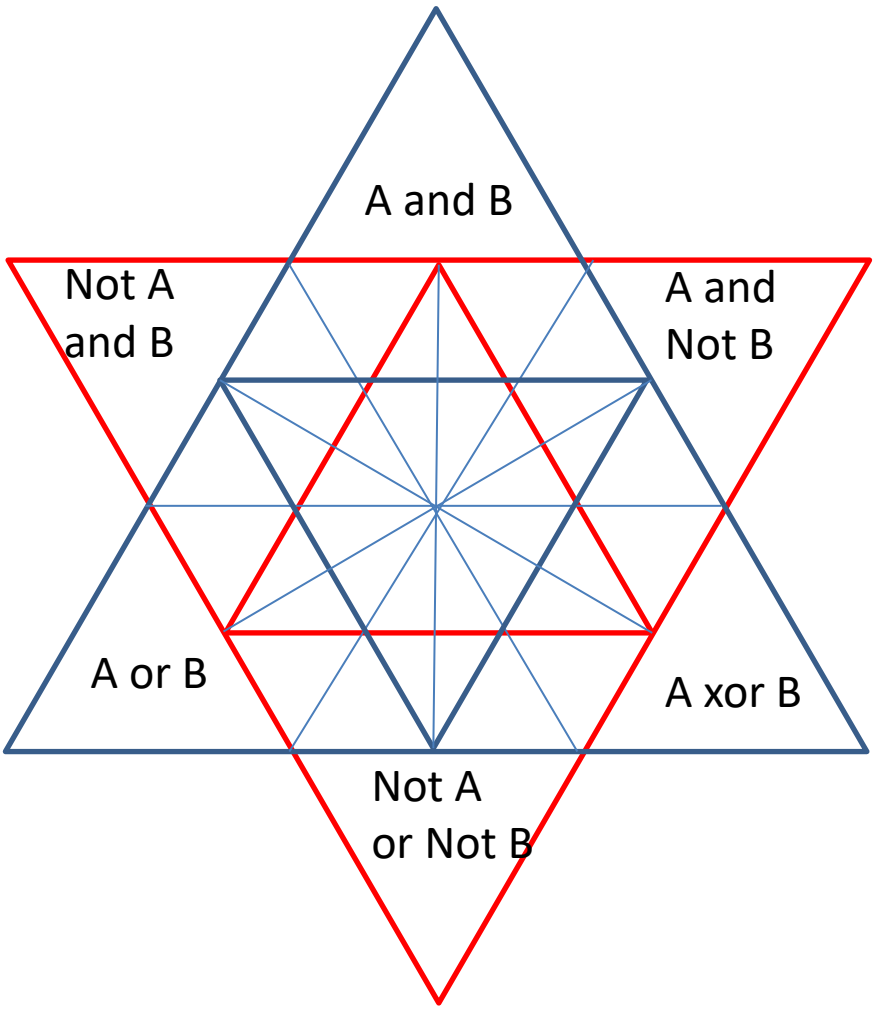


A - positive point of view
B - negative point of view
C - neutral point of view
D - objective point of view
A (AND, OR, XOR, NOT, NAND, NOR, XNOR) C ->a1
A(....)D->a2
a1(....)a2->A1
B (....) C->b1
B (....) D ->b2
b1(....)b2->B1
A1 (AND, OR, XOR, NOT, NAND, NOR, XNOR) B1 ->Desition

GROK

Let's calculate whether Russia should engage in peace negotiations based on the updated model provided. The structure is as follows:A: Positive point of viewB: Negative point of viewC: Neutral point of viewD: Objective point of viewa1 = A (logical operation) Ca2 = A (logical operation) DA1 = a1 (logical operation) a2b1 = B (logical operation) Cb2 = B (logical operation) DB1 = b1 (logical operation) b2Decision = A1 (logical operation) B1Since specific logical operations aren't predefined, I'll select a combination that reflects a coherent decision-making process and explain the choices:A AND C → a1: Positive view must align with neutral view.A OR D → a2: Positive view or objective reality is sufficient.a1 XOR a2 → A1: Divergence between alignment and broader possibility.B AND C → b1: Negative view must align with neutral view.B OR D → b2: Negative view or objective reality is sufficient.b1 XOR b2 → B1: Divergence between alignment and broader possibility.A1 OR B1 → Decision: Either the positive or negative combined reasoning justifies action.This setup balances strict alignment (AND) with broader considerations (OR), using XOR to highlight internal tensions, and OR for a final inclusive decision.Step 1: Define the Perspectives (as of April 6, 2025)A - Positive Point of View (Reasons Russia Should Negotiate)Strategic consolidation of gains (e.g., eastern Ukraine, Crimea).Economic relief from sanctions and war costs.Domestic support for peace (e.g., 54% favoring talks in late 2024 polls).B - Negative Point of View (Reasons Russia Should Not Negotiate)Maximalist goals (e.g., Ukraine's neutrality, full territorial recognition) unmet.Current battlefield momentum favors continued pressure.Risk of losing leverage or making unwanted concessions.C - Neutral Point of ViewTalks are inevitable eventually, but timing and terms remain unclear.External actors (e.g., China, U.S.) push for talks, with uncertain impact.Neither side shows clear willingness to compromise yet.D - Objective Point of ViewMilitary: ~20% of Ukraine controlled, but high losses (personnel, equipment).Economy: Growth persists (~4% in 2024), but sanctions and isolation threaten long-term stability.Diplomacy: Putin demands maximalist terms; U.S. (e.g., Trump's efforts) pushes talks.Step 2: Compute a1 = A AND CA: Negotiations could secure gains, ease economic strain, and align with public sentiment.C: Talks may happen eventually, but timing and outcome are uncertain.A AND C: True only if positive reasons align with neutral inevitability.Result: False. The positive push for immediate gains doesn't fully match the neutral uncertainty about timing and terms.Step 3: Compute a2 = A OR DA: Negotiations could benefit Russia strategically and economically.D: Objective costs (military, economic) suggest talks might be pragmatic.A OR D: True if either positive benefits or objective realities hold.Result: True. Both potential gains and objective pressures support negotiations.Step 4: Compute A1 = a1 XOR a2a1 = False, a2 = True.XOR: True if one is true and the other false.Result: True (False XOR True = True). The positive-neutral misalignment contrasts with broader positive-objective support.Step 5: Compute b1 = B AND CB: No negotiations due to unmet goals, battlefield advantage, and concession risks.C: Talks are inevitable but uncertain in timing and terms.B AND C: True only if negative reasons align with neutral stance.Result: False. The negative view (avoid talks now) clashes with neutral inevitability.Step 6: Compute b2 = B OR DB: Continuing the war preserves leverage and goals.D: Objective costs suggest talks could be rational.B OR D: True if either negative stance or objective reality holds.Result: True. Either maintaining pressure or acknowledging costs is valid.Step 7: Compute B1 = b1 XOR b2b1 = False, b2 = True.XOR: True if one is true and the other false.Result: True (False XOR True = True). The negative-neutral mismatch contrasts with broader negative-objective viability.Step 8: Compute Decision = A1 OR B1A1 = True, B1 = True.OR: True if either A1 or B1 is true.Result: True (True OR True = True). Both combined positive and negative analyses suggest a decision to act.Interpretation of DecisionDecision = True: Russia should engage in peace negotiations. The model shows that despite internal tensions (XOR results), both the positive (A1) and negative (B1) perspectives, when combined with neutral and objective views, lean toward talks. The OR operation at the end indicates that either the potential benefits (A1) or the costs of avoidance (B1) justify negotiation.Final Answer (in Russian)По модели:a1 = A AND C = False (позитив и нейтралитет не совпадают).a2 = A OR D = True (позитив или объективность поддерживают переговоры).A1 = a1 XOR a2 = True (различие между a1 и a2).b1 = B AND C = False (негатив и нейтралитет не совпадают).b2 = B OR D = True (негатив или объективность имеют вес).B1 = b1 XOR b2 = True (различие между b1 и b2).Решение = A1 OR B1 = True (True OR True = True).России следует идти на переговоры о мире. Модель показывает, что несмотря на противоречия внутри позитивных и негативных взглядов, сочетание с нейтральными и объективными факторами (потери, экономика, неизбежность) указывает на целесообразность переговоров на 6 апреля 2025 года.

36 logic operation

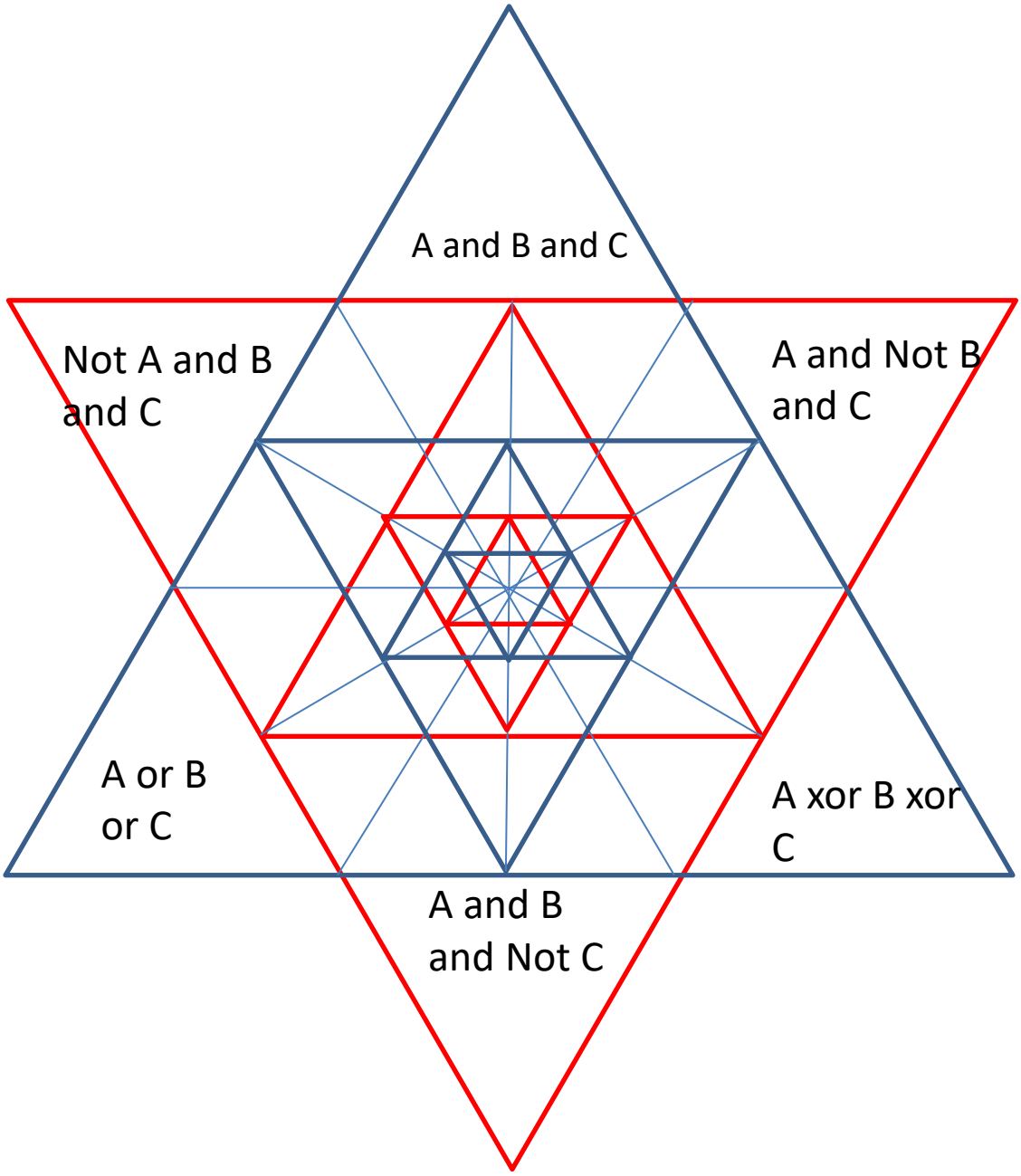


Fedorchenko Mikhail Valerevich

№	Вывод (A,B)	00	01	10	11	Формула (логическое выражение)
1.1	AND B	0	0	0	1	A AND B
1.2	A OR B	0	1	1	1	A OR B
1.3	A XOR B	0	1	1	0	A XOR B
1.4	NOT (A OR B)	1	0	0	0	NOT (A OR B)
1.5	NOT A AND B	0	1	0	0	NOT A AND B
1.6	A AND NOT B	0	0	1	0	A AND NOT B
2.1	A NOR B	1	0	0	0	NOT (A OR B)
2.2	A OR (NOT B)	1	0	1	1	A OR NOT B
2.3	AXNOR B	1	0	0	1	NOT (A XOR B)
2.4	B	0	1	0	1	B
2.5	(NOT A) OR B	1	1	0	1	NOT A OR B
2.6	(A XOR B) OR ((NOT A) AND (NOT B))	1	1	1	0	(A XOR B) OR (NOT A AND NOT B)
3.1	(A AND (NOT B)) OR ((NOT A) AND B)	0	1	1	0	(A AND NOT B) OR (NOT A AND B)
3.2	NOT (A AND B)	1	1	1	0	NOT (A AND B)
3.3	((NOT A) AND B) OR (A AND (NOT B))	0	0	1	0	(NOT A AND B) OR (A AND NOT B)
3.4	NOT B	1	0	1	0	NOT B
3.5	A NAND (NOT B)	1	1	0	0	NOT (A AND NOT B)
3.6	(A OR B) AND NOT (A AND B)	0	1	1	0	(A OR B) AND NOT (A AND B)
4.1	A AND B AND NOT (A XOR B)	0	0	0	1	A AND B AND NOT (A XOR B)
4.2	A OR NOT B	1	0	1	1	A OR NOT B
	((NOT A) OR B) AND (A OR (NOT					

4.3	B))	1	0	0	0	(NOT A OR B) AND (A OR NOT B)
4.4	B OR (NOT A)	1	1	0	1	B OR NOT A
4.5	NOT (A OR B)	1	0	0	0	NOT (A OR B)
4.6	NOT (A AND B) OR (A XOR B)	1	1	1	0	NOT (A AND B) OR (A XOR B)
5.1	(A OR B) OR (A XOR B)	0	1	1	1	(A OR B) OR (A XOR B)
5.2	(A AND NOT B) OR ((NOT A) AND B)	0	1	1	0	(A AND NOT B) OR (NOT A AND B)
5.3	(B AND (NOT A)) OR (A AND B)	0	0	0	1	(B AND NOT A) OR (A AND B)
5.4	(NOT A) AND (NOT B)	1	0	0	0	NOT A AND NOT B
5.5	(A AND B) OR ((NOT A) AND (NOT B))	1	0	0	1	(A AND B) OR (NOT A AND NOT B)
5.6	NOT (A XOR B)	1	0	0	1	NOT (A XOR B)
6.1	(A AND (NOT B)) OR ((NOT A) AND B)	0	1	1	0	(A AND NOT B) OR (NOT A AND B)
6.2	(NOT B) OR A	1	0	1	1	NOT B OR A
6.3	B AND ((NOT A) OR B)	0	1	0	1	B AND (NOT A OR B)
6.4	A XOR (NOT B)	1	1	0	0	A XOR NOT B
6.5	(A OR B) AND NOT (A AND (NOT B))	0	1	0	1	(A OR B) AND NOT (A AND NOT B)
6.6	NOT (A AND (NOT B))	1	1	0	1	NOT (A AND NOT B)

74+12 logical operation



generation full list 72+14((18*4(via intersection and intersection with vertices calculation))+14) logic opiration in diagramm

Fedorchenko Mikhail Valerevich

Enumerated Logical Operations

For three variables (A, B, C), there are $2^3 = 8$ basic truth value combinations. The operations can include:

- A and B and C
- Not A and B and C
- A and Not B and C
- Not A and Not B and C
- A and B and Not C
- Not A and B and Not C
- A and Not B and Not C
- Not A and Not B and Not C

Next, include OR operations:
9. A or B or C
10. Not A or B or C
11. A or Not B or C
12. Not A or Not B or C
13. A or B or Not C
14. Not A or B or Not C
15. A or Not B or Not C
16. Not A or Not B or Not C

Now, include XOR operations (exclusive OR for three variables is more complex, but we can list pairwise and triple XOR):
17. A xor B xor C
18. Not A xor B xor C
19. A xor Not B xor C
20. Not A xor Not B xor C
21. A xor B xor Not C
22. Not A xor B xor Not C
23. A xor Not B xor Not C
24. Not A xor Not B xor Not C

Add mixed operations (e.g., combining AND and NOT, OR and NOT):
25. A and (B or C)

26. Not A and (B or C)
27. A and (Not B or C)
28. Not A and (Not B or C)
29. A and (B or Not C)
30. Not A and (B or Not C)
31. A and (Not B or Not C)
32. Not A and (Not B or Not C)
33. (A or B) and C
34. (Not A or B) and C
35. (A or Not B) and C
36. (Not A or Not B) and C
37. (A or B) and Not C
38. (Not A or B) and Not C
39. (A or Not B) and Not C
40. (Not A or Not B) and Not C

Continue with additional intersections and combinations implied by the nested structure (e.g., partial overlaps):

41. A and Not B and C
42. Not A and B and C (repeated for clarity)
43. A or B and Not C
44. Not A or B and Not C
45. A and B or C
46. Not A and B or C
47. A and Not B or C
48. Not A and Not B or C
49. A xor B and C
50. Not A xor B and C
51. A xor Not B and C
52. Not A xor Not B and C
53. A xor B and Not C
54. Not A xor B and Not C
55. A xor Not B and Not C
56. Not A xor Not B and Not C

To reach 86 operations, continue with more complex combinations and intersections:

57. (A and B) or (B and C)
58. (Not A and B) or (B and C)
59. (A and Not B) or (B and C)

60. (A and B) or (Not B and C)
61. (Not A and B) or (Not B and C)
62. (A and Not B) or (Not B and C)
63. (A and B) or (B and Not C)
64. (Not A and B) or (B and Not C)
65. (A and Not B) or (B and Not C)
66. (A and B) or (Not B and Not C)
67. (Not A and B) or (Not B and Not C)
68. (A and Not B) or (Not B and Not C)
69. (A or B) and (B or C)
70. (A or Not B) and (B or C)
71. (A or B) and (Not B or C)
72. (A or Not B) and (Not B or C)
73. (A or B) and (B or Not C)
74. (A or Not B) and (B or Not C)
75. (A or B) and (Not B or Not C)
76. (A or Not B) and (Not B or Not C)
77. (A and B) xor C
78. (Not A and B) xor C
79. (A and Not B) xor C
80. (Not A and Not B) xor C
81. A xor (B and C)
82. Not A xor (B and C)
83. A xor (Not B and C)
84. Not A xor (Not B and C)
85. A xor (B and Not C)
86. Not A xor (B and Not C)

A logical continuation innovation of the project.

The document "36 logic operation.pdf" primarily lists basic logical operations (e.g., A and B, A or B, Not A, Not B) with repetitive mentions of "36 logic operation" and "366 logic operation," alongside some seemingly incomplete or redundant entries (e.g., "A and"). The presence of "36" and "366" suggests a possible reference to a specific framework, system, or project identifier, but the document lacks context or detailed explanations. The URLs on pages 2 and 3 appear malformed or incomplete, making it impossible to retrieve additional context without further clarification.

Given your request for "a logical continuation innovation of the project," I'll assume the project relates to logical operations, possibly in a computational, programming, or systems design context. Below is a proposed innovation to extend a project centered on logical operations:

Logical Continuation Innovation: Enhanced Logical Operation Framework

To innovate a project based on logical operations like those listed (AND, OR, NOT), I propose developing a **Dynamic Logic Optimization System (DLOS)**. This system would extend the basic logical operations into a scalable, adaptive framework for real-world applications in areas like artificial intelligence, circuit design, or decision-making systems. Here's the concept:

1. Core Innovation: Adaptive Logic Processing

- **Concept**: Create a system that dynamically selects and optimizes logical operations based on input complexity, resource constraints, and desired outcomes. Unlike static logic gates, DLOS would use machine learning to predict the most efficient combination of logical operations for a given task.
- **Example**: For a circuit design task, DLOS could analyze input signals (A, B) and decide whether "A AND B" or "A OR B" minimizes power consumption or latency, adapting in real-time to changing conditions.

2. Features of DLOS

- **Modular Logic Library**: Expand beyond basic AND, OR, NOT to include XOR, NAND, NOR, and complex combinations, with customizable "36 logic operation" modules (assuming "36" refers to a specific set of operations or a project code).
- **Context-Aware Optimization**: Integrate context from the environment (e.g., hardware limits, data patterns) to prioritize operations. For instance, if "366 logic operation" implies a higher-order operation set, DLOS could switch between "36" and "366" modes based on computational needs.
- **Error Correction**: Introduce self-correcting logic to handle incomplete operations (e.g., fixing "A and" by inferring the missing operand or defaulting to a predefined rule).
- **Scalability**: Apply the system to micro-level (e.g., chip design) or macro-level (e.g., decision trees in AI) applications.

3. Implementation Steps

- **Step 1: Define the "36/366 Logic Operation" Scope**: Clarify whether "36" and "366" refer to specific operation sets, project versions, or numerical constraints. For example, "36" could imply 36 predefined logic rules, while "366" might indicate an expanded set.
- **Step 2: Build a Logic Simulation Engine**: Create a software platform to simulate and test combinations of logical operations, using the document's listed operations (A AND B, A OR B, etc.) as a foundation.
- **Step 3: Integrate AI**: Use machine learning to analyze input data and select optimal logic operations dynamically. For example, a neural network could learn when to prioritize NOT A over A AND B based on historical performance.
- **Step 4: Hardware/Software Integration**: Develop interfaces for DLOS to work with hardware (e.g., FPGA for circuit design) or software (e.g., decision-making algorithms in Python).
- **Step 5: Testing and Iteration**: Test DLOS on real-world problems, such as optimizing Boolean expressions in circuit design or enhancing AI decision-making efficiency.

4. Potential Applications

- **Circuit Design**: Optimize logic gate configurations for low-power, high-speed microchips.

- **AI Decision Systems**: Enhance AI models by dynamically adjusting logical rules for tasks like natural language processing or game strategy.
- **Automation**: Improve industrial control systems by adapting logical operations to sensor inputs in real-time.
- **Data Processing**: Streamline database queries by optimizing logical conditions for faster execution.

5. Addressing Document Limitations

- **Repetition and Incompleteness**: The document's repetitive and incomplete entries (e.g., "A and") suggest a need for error-checking mechanisms in DLOS to handle malformed inputs.
- **Ambiguity of "36/366"**: Without specific context, DLOS would include a flexible framework to accommodate varying interpretations (e.g., 36 as a version number, operation count, or symbolic reference).
- **URLs**: The malformed URLs (e.g., "https://x.com/igrack/...") indicate a need for robust external data integration. DLOS could include a web-scraping module to validate and extract relevant information from similar links, if provided.

Next Steps

To refine this innovation:

- **Clarify "36/366 Logic Operation"**: Provide details on what these terms represent in the project context.
- **Specify Project Goals**: Are you focusing on hardware, software, AI, or another domain?
- **Validate URLs**: If you have correct URLs or additional documents, I can analyze them to provide more tailored suggestions.
- **Implementation Details**: Would you like a code snippet, diagram, or specific technical approach (e.g., Python for simulation, Verilog for hardware)?

Please provide more context or confirm the direction, and I'll tailor the innovation further!

Clarify "36/366 Logic Operation"

The terms "36 logic operation" and "366 logic operation" in the provided document remain ambiguous due to the lack of explicit context or definitions within the text. Based on the document's content, which repeatedly lists basic logical operations (e.g., A AND B, A OR B, NOT A, NOT B) alongside these terms, and considering the search results on computational logic frameworks, here's a clarified interpretation:

Possible Meanings of "36/366 Logic Operation"

- **Project or Version Identifiers**:
  - "36" and "366" could be codenames, version numbers, or identifiers for specific sets of logical operations or computational frameworks within a project. For example, they might refer to different iterations or configurations of a logic-based system, such as a circuit design, algorithm, or logic programming module.
  - The repetition in the document suggests "36" and "366" might distinguish between two related but distinct sets of operations or rules, possibly with "366" being an extended or modified version of "36."
- **Number of Operations or Rules**:
  - The numbers might indicate the count of logical operations or rules in a system. For instance, "36 logic operation" could imply a framework with 36 predefined logical operations (e.g., combinations of AND, OR, NOT, etc.), while "366" might represent an expanded set.
  - This aligns with computational logic frameworks, where systems like Boolean algebra or logic programming (e.g., Prolog) define sets of operations or clauses to solve computational problems.
- **Symbolic or Contextual Reference**:
  - The numbers could be symbolic or tied to a specific context not explicitly stated, such as a reference to a hardware specification, a dataset size, or a temporal factor (e.g., 36/366 resembling days in a year, though less likely). The document's structure doesn't support a clear temporal link, so this is speculative.

- Alternatively, they might refer to specific logical formalisms or axioms, as seen in computational logic systems like clarithmetics, which define computational problems with precise rules.

- **Error or Placeholder**:
  - The document's repetitive and incomplete entries (e.g., "A and") and the presence of malformed URLs suggest possible errors in transcription or placeholders for a more detailed system description. "36" and "366" might be artifacts of incomplete documentation or typographical errors (e.g., "366" as a misentry for "36").

Contextual Analysis

- **Document Content**: The document lists basic Boolean operations (AND, OR, NOT) repeatedly, with "36 logic operation" and "366 logic operation" interspersed. This suggests a focus on fundamental logic operations, possibly within a computational or engineering context like circuit design or formal verification.
- **Search Results**: Computational logic involves using logical formalisms to perform computations, often in circuit design, software verification, or AI. The terms "36" and "366" don't directly match any standard framework in the results, but they could align with custom or project-specific logic sets, similar to how clarithmetics define problem-specific rules.
- **Potential Misinterpretation**: The numbers might not refer to operations themselves but to a system's parameters, such as the number of logic gates, clauses in a logic program, or states in a computational model. Without further context, this remains uncertain.

Logical Continuation Innovation (Refined)

Given the ambiguity, a logical continuation of the project could involve:

- **Standardizing the Framework**: Define "36" and "366" as specific sets of logical operations or rules within a computational logic system. For example, "36" could be a core set of Boolean operations, while "366" includes advanced operations (e.g., NAND, NOR, XOR) or interactive rules like those in computability logic.
- **Developing a Specification**: Create a formal specification for the "36/366" framework, clarifying the operations, their syntax, and their application (e.g., in circuit design or logic programming).
- **Automation and Verification**: Build a tool to automate the application of these operations, using formal verification to ensure correctness, as seen in systems like Isabelle or ACL2.

Next Steps

To provide a more precise clarification:

- **Provide Context**: Share any additional details about the project (e.g., is it related to hardware, software, AI, or a specific logic system?).
- **Confirm Numbers**: Are "36" and "366" intentional, or could they be errors or placeholders? Do they relate to a specific count, version, or system component?
- **Fix URLs**: If you have corrected versions of the URLs (e.g., from pages 2 and 3), I can analyze them for further context.
- **Specify Domain**: Is the project focused on Boolean algebra, logic programming, formal verification, or another area?

Without additional details, I recommend treating "36/366 logic operation" as identifiers for distinct sets of logical rules or operations within a computational framework, with "366" potentially being an extended or revised version of "36." Please provide more information to refine this further!

366 logic operation

The term "366 logic operation" in the provided document remains ambiguous due to insufficient context, but I can offer a refined interpretation based on the document's content, the previous clarification of "36/366 logic operation," and the web search results on computational logic. The document repeatedly lists basic logical operations (e.g., A AND B, A OR B, NOT A, NOT B) alongside "36 logic operation" and "366 logic operation," suggesting that "366" may refer to an extended or distinct set of logical operations, a project identifier, or a specific framework within a computational logic context.

Clarification of "366 Logic Operation"

Given the lack of a direct definition in the document or search results, here are the most likely interpretations:

- **Extended Set of Logical Operations**:
  - "366 logic operation" could denote an expanded set of logical operations compared to "36 logic operation." For example, while "36" might refer to a core set of 36 operations (e.g., combinations of AND, OR, NOT, and derived operations like NAND, NOR, XOR), "366" could represent a larger or more complex set, possibly including additional operators, temporal logic constructs, or application-specific rules.
  - In computational logic, operations can extend beyond basic Boolean operators to include temporal operators (e.g., "Next," "Globally," "Until" in Computation Tree Logic) or constraint-based rules in Constraint Logic Programming (CLP).
  - Example: If "36" covers basic Boolean operations, "366" might include advanced constructs like those in CLP, where rules handle constraints (e.g., $X \leq Y$) or propagate logical consequences.
- **Project or Version Identifier**:
  - "366" may be a codename, version number, or identifier for a specific computational logic framework or project iteration. The document's repetition of "366 logic operation" alongside "36 logic operation" suggests a possible evolution or variant, with "366" being a more advanced or comprehensive version.
  - This aligns with computational logic's history, where terms like "computational logic" were reused for projects like Compulog-II or the ACM Transactions on Computational Logic to denote extensions of earlier work.
- **Symbolic or Numerical Reference**:
  - The number "366" might symbolize a specific parameter, such as the number of rules, states, or components in a system. For instance, it could refer to 366 logical rules in a logic programming system, 366 logic gates in a circuit design, or a dataset with 366 entries (e.g., resembling days in a leap year, though this is speculative).
  - Alternatively, it could be a placeholder or error, as the document contains incomplete entries (e.g., "A and") and malformed URLs, suggesting possible transcription issues.
- **Domain-Specific Application**:
  - In computational logic, operations are applied in contexts like formal verification, circuit design, or AI. "366 logic operation" might refer to a specialized set of operations for a specific application, such as verifying a system with 366 states or optimizing a circuit with 366 gates.
  - For example, in circuit design, logical operations (AND, OR, NOT, etc.) are implemented as logic gates, and a system might use a specific number of gates or rules, potentially denoted by "366."

Logical Continuation Innovation for "366 Logic Operation"

Building on the assumption that "366 logic operation" refers to an extended or specialized set of logical operations within a computational framework, here's a proposed innovation to advance the project:

Enhanced Logic Optimization Framework (ELOF-366)

Concept: Develop a system that leverages an extended set of 366 logical operations (or rules) to optimize computational tasks in areas like circuit design, formal verification, or AI-driven decision-making. This framework would build on the basic operations listed in the document (AND, OR, NOT) and incorporate advanced constructs for greater flexibility and efficiency.

Key Features:

- **Comprehensive Operation Set**:
  - Define "366 logic operation" as a library of 366 operations, including:
    - Basic Boolean operators (AND, OR, NOT, NAND, NOR, XOR, XNOR).
    - Temporal operators (e.g., Next, Globally, Until) for systems like Computation Tree Logic (CTL).
    - Constraint-based rules (e.g., $X \leq Y$) for Constraint Logic Programming.
    - Custom operations tailored to specific applications (e.g., bit-masking in circuit design or probabilistic logic in AI).
  - Example: A rule like "IF (A AND B) OR (NOT C)" could be one of the 366 operations, optimized for specific hardware or software constraints.

- **Dynamic Operation Selection**:
  - Use machine learning to dynamically select the most efficient subset of the 366 operations based on input data, system constraints (e.g., power, speed), and task requirements.
  - Example: In a low-power circuit, ELOF-366 might prioritize NAND gates (functionally complete) over AND/OR combinations to minimize gate count.
- **Formal Verification Integration**:
  - Incorporate tools like Isabelle or model checkers to verify the correctness of the 366 operations in applications like software or hardware design, ensuring no crashes or failures.
  - Example: Verify that a circuit using the "366" operations satisfies safety properties (e.g., no division by zero).
- **Scalability and Modularity**:
  - Design ELOF-366 to scale from small systems (e.g., 8-bit processors) to large systems (e.g., 64-bit architectures or AI models), adapting the operation set dynamically.
  - Allow modular additions, so new operations can be integrated into the "366" framework as needed.
- **Error Handling**:
  - Address document issues (e.g., incomplete "A and" entries) by including robust parsing and error-correction mechanisms to handle malformed logical expressions.

Implementation Steps:

- **Define the 366 Operations**: Create a formal specification for the 366 operations, combining Boolean, temporal, and constraint-based rules. If "366" is a project identifier, catalog the operations under this version.
- **Build a Simulation Platform**: Develop a software tool (e.g., in Python or Verilog) to simulate and test the 366 operations, ensuring compatibility with hardware (e.g., FPGAs) or software (e.g., logic programming systems).
- **Optimize with AI**: Train an AI model to select optimal operations from the 366-set based on performance metrics (e.g., execution time, power consumption).
- **Test in Applications**: Apply ELOF-366 to real-world scenarios, such as optimizing a digital circuit, verifying an operating system kernel, or enhancing AI decision-making.
- **Document and Standardize**: Create clear documentation to avoid ambiguities like those in the original document, specifying each operation's syntax and semantics.

Potential Applications:

- **Circuit Design**: Optimize logic gate configurations for high-speed, low-power integrated circuits using the 366 operations.
- **Formal Verification**: Prove system correctness in safety-critical applications (e.g., pacemakers, aircraft control systems) using the extended operation set.
- **AI and Logic Programming**: Enhance logic programming systems (e.g., Prolog, CLP) with the 366 operations for more expressive and efficient rule-based reasoning.
- **Search Algorithms**: Improve search efficiency by optimizing logical conditions within the 366 framework.

Addressing Document Limitations

- **Ambiguity of "366"**: Without a clear definition, ELOF-366 assumes "366" as an extended operation set. If you can confirm whether it's a version number, rule count, or other parameter, I can refine the framework.
- **Incomplete Entries**: The document's errors (e.g., "A and") are mitigated by including error-checking in ELOF-366 to handle malformed inputs.
- **Malformed URLs**: The URLs in the document (e.g., "https://ix.com/igrack/...") are invalid, preventing further context analysis. If you have corrected URLs or additional documents, please provide them.

Next Steps

To further refine the innovation:

- **Confirm "366" Meaning**: Is it a count of operations, a version number, or a domain-specific reference (e.g., circuit gates, logic rules)?

- **Specify Domain**: Is the project focused on hardware (e.g., logic gates), software (e.g., logic programming), or another area like AI or formal verification?
- **Provide Additional Context**: Share any related project details, specifications, or corrected URLs to clarify the scope.
- **Request Specific Output**: Would you like a code snippet (e.g., Python for simulating the 366 operations), a truth table for sample operations, or a diagram of the framework?

432 logic operation

Group	Expression
Group 1	(A) and (A) and (A)
Group 1	(A) and (A) and (B)
Group 1	(A) and (A) and (C)
Group 1	(A) and (A) and (not A)
Group 1	(A) and (A) and (not B)
Group 1	(A) and (A) and (not C)
Group 1	(A) and (B) and (A)
Group 1	(A) and (B) and (B)
Group 1	(A) and (B) and (C)
Group 1	(A) and (B) and (not A)
Group 1	(A) and (B) and (not B)
Group 1	(A) and (B) and (not C)
Group 1	(A) and (C) and (A)
Group 1	(A) and (C) and (B)
Group 1	(A) and (C) and (C)
Group 1	(A) and (C) and (not A)
Group 1	(A) and (C) and (not B)
Group 1	(A) and (C) and (not C)
Group 1	(A) and (not A) and (A)
Group 1	(A) and (not A) and (B)
Group 1	(A) and (not A) and (C)
Group 1	(A) and (not A) and (not A)
Group 1	(A) and (not A) and (not B)
Group 1	(A) and (not A) and (not C)
Group 1	(A) and (not B) and (A)
Group 1	(A) and (not B) and (B)
Group 1	(A) and (not B) and (C)
Group 1	(A) and (not B) and (not A)
Group 1	(A) and (not B) and (not B)
Group 1	(A) and (not B) and (not C)
Group 1	(A) and (not C) and (A)
Group 1	(A) and (not C) and (B)
Group 1	(A) and (not C) and (C)
Group 1	(A) and (not C) and (not A)
Group 1	(A) and (not C) and (not B)
Group 1	(A) and (not C) and (not C)
Group 1	(B) and (A) and (A)
Group 1	(B) and (A) and (B)
Group 1	(B) and (A) and (C)
Group 1	(B) and (A) and (not A)
Group 1	(B) and (A) and (not B)
Group 1	(B) and (A) and (not C)
Group 1	(B) and (B) and (A)
Group 1	(B) and (B) and (B)
Group 1	(B) and (B) and (C)
Group 1	(B) and (B) and (not A)

Group 1 (B) and (B) and (not B)
Group 1 (B) and (B) and (not C)
Group 1 (B) and (C) and (A)
Group 1 (B) and (C) and (B)
Group 1 (B) and (C) and (C)
Group 1 (B) and (C) and (not A)
Group 1 (B) and (C) and (not B)
Group 1 (B) and (C) and (not C)
Group 1 (B) and (not A) and (A)
Group 1 (B) and (not A) and (B)
Group 1 (B) and (not A) and (C)
Group 1 (B) and (not A) and (not A)
Group 1 (B) and (not A) and (not B)
Group 1 (B) and (not A) and (not C)
Group 1 (B) and (not B) and (A)
Group 1 (B) and (not B) and (B)
Group 1 (B) and (not B) and (C)
Group 1 (B) and (not B) and (not A)
Group 1 (B) and (not B) and (not B)
Group 1 (B) and (not B) and (not C)
Group 1 (B) and (not C) and (A)
Group 1 (B) and (not C) and (B)
Group 1 (B) and (not C) and (C)
Group 1 (B) and (not C) and (not A)
Group 1 (B) and (not C) and (not B)
Group 1 (B) and (not C) and (not C)
Group 2 (A) or (A) or (A)
Group 2 (A) or (A) or (B)
Group 2 (A) or (A) or (C)
Group 2 (A) or (A) or (not A)
Group 2 (A) or (A) or (not B)
Group 2 (A) or (A) or (not C)
Group 2 (A) or (B) or (A)
Group 2 (A) or (B) or (B)
Group 2 (A) or (B) or (C)
Group 2 (A) or (B) or (not A)
Group 2 (A) or (B) or (not B)
Group 2 (A) or (B) or (not C)
Group 2 (A) or (C) or (A)
Group 2 (A) or (C) or (B)
Group 2 (A) or (C) or (C)
Group 2 (A) or (C) or (not A)
Group 2 (A) or (C) or (not B)
Group 2 (A) or (C) or (not C)
Group 2 (A) or (not A) or (A)
Group 2 (A) or (not A) or (B)
Group 2 (A) or (not A) or (C)

Group 2 (A) or (not A) or (not A)
Group 2 (A) or (not A) or (not B)
Group 2 (A) or (not A) or (not C)
Group 2 (A) or (not B) or (A)
Group 2 (A) or (not B) or (B)
Group 2 (A) or (not B) or (C)
Group 2 (A) or (not B) or (not A)
Group 2 (A) or (not B) or (not B)
Group 2 (A) or (not B) or (not C)
Group 2 (A) or (not C) or (A)
Group 2 (A) or (not C) or (B)
Group 2 (A) or (not C) or (C)
Group 2 (A) or (not C) or (not A)
Group 2 (A) or (not C) or (not B)
Group 2 (A) or (not C) or (not C)
Group 2 (B) or (A) or (A)
Group 2 (B) or (A) or (B)
Group 2 (B) or (A) or (C)
Group 2 (B) or (A) or (not A)
Group 2 (B) or (A) or (not B)
Group 2 (B) or (A) or (not C)
Group 2 (B) or (B) or (A)
Group 2 (B) or (B) or (B)
Group 2 (B) or (B) or (C)
Group 2 (B) or (B) or (not A)
Group 2 (B) or (B) or (not B)
Group 2 (B) or (B) or (not C)
Group 2 (B) or (C) or (A)
Group 2 (B) or (C) or (B)
Group 2 (B) or (C) or (C)
Group 2 (B) or (C) or (not A)
Group 2 (B) or (C) or (not B)
Group 2 (B) or (C) or (not C)
Group 2 (B) or (not A) or (A)
Group 2 (B) or (not A) or (B)
Group 2 (B) or (not A) or (C)
Group 2 (B) or (not A) or (not A)
Group 2 (B) or (not A) or (not B)
Group 2 (B) or (not A) or (not C)
Group 2 (B) or (not B) or (A)
Group 2 (B) or (not B) or (B)
Group 2 (B) or (not B) or (C)
Group 2 (B) or (not B) or (not A)
Group 2 (B) or (not B) or (not B)
Group 2 (B) or (not B) or (not C)
Group 2 (B) or (not C) or (A)
Group 2 (B) or (not C) or (B)

Group 2	(B) or (not C) or (C)
Group 2	(B) or (not C) or (not A)
Group 2	(B) or (not C) or (not B)
Group 2	(B) or (not C) or (not C)
Group 3	((A) xor (A)) xor (A)
Group 3	((A) xor (A)) xor (B)
Group 3	((A) xor (A)) xor (C)
Group 3	((A) xor (A)) xor (not A)
Group 3	((A) xor (A)) xor (not B)
Group 3	((A) xor (A)) xor (not C)
Group 3	((A) xor (B)) xor (A)
Group 3	((A) xor (B)) xor (B)
Group 3	((A) xor (B)) xor (C)
Group 3	((A) xor (B)) xor (not A)
Group 3	((A) xor (B)) xor (not B)
Group 3	((A) xor (B)) xor (not C)
Group 3	((A) xor (C)) xor (A)
Group 3	((A) xor (C)) xor (B)
Group 3	((A) xor (C)) xor (C)
Group 3	((A) xor (C)) xor (not A)
Group 3	((A) xor (C)) xor (not B)
Group 3	((A) xor (C)) xor (not C)
Group 3	((A) xor (not A)) xor (A)
Group 3	((A) xor (not A)) xor (B)
Group 3	((A) xor (not A)) xor (C)
Group 3	((A) xor (not A)) xor (not A)
Group 3	((A) xor (not A)) xor (not B)
Group 3	((A) xor (not A)) xor (not C)
Group 3	((A) xor (not B)) xor (A)
Group 3	((A) xor (not B)) xor (B)
Group 3	((A) xor (not B)) xor (C)
Group 3	((A) xor (not B)) xor (not A)
Group 3	((A) xor (not B)) xor (not B)
Group 3	((A) xor (not B)) xor (not C)
Group 3	((A) xor (not C)) xor (A)
Group 3	((A) xor (not C)) xor (B)
Group 3	((A) xor (not C)) xor (C)
Group 3	((A) xor (not C)) xor (not A)
Group 3	((A) xor (not C)) xor (not B)
Group 3	((A) xor (not C)) xor (not C)
Group 3	((B) xor (A)) xor (A)
Group 3	((B) xor (A)) xor (B)
Group 3	((B) xor (A)) xor (C)
Group 3	((B) xor (A)) xor (not A)
Group 3	((B) xor (A)) xor (not B)
Group 3	((B) xor (A)) xor (not C)
Group 3	((B) xor (B)) xor (A)

Group 3	((B) xor (B)) xor (B)
Group 3	((B) xor (B)) xor (C)
Group 3	((B) xor (B)) xor (not A)
Group 3	((B) xor (B)) xor (not B)
Group 3	((B) xor (B)) xor (not C)
Group 3	((B) xor (C)) xor (A)
Group 3	((B) xor (C)) xor (B)
Group 3	((B) xor (C)) xor (C)
Group 3	((B) xor (C)) xor (not A)
Group 3	((B) xor (C)) xor (not B)
Group 3	((B) xor (C)) xor (not C)
Group 3	((B) xor (not A)) xor (A)
Group 3	((B) xor (not A)) xor (B)
Group 3	((B) xor (not A)) xor (C)
Group 3	((B) xor (not A)) xor (not A)
Group 3	((B) xor (not A)) xor (not B)
Group 3	((B) xor (not A)) xor (not C)
Group 3	((B) xor (not B)) xor (A)
Group 3	((B) xor (not B)) xor (B)
Group 3	((B) xor (not B)) xor (C)
Group 3	((B) xor (not B)) xor (not A)
Group 3	((B) xor (not B)) xor (not B)
Group 3	((B) xor (not B)) xor (not C)
Group 3	((B) xor (not C)) xor (A)
Group 3	((B) xor (not C)) xor (B)
Group 3	((B) xor (not C)) xor (C)
Group 3	((B) xor (not C)) xor (not A)
Group 3	((B) xor (not C)) xor (not B)
Group 3	((B) xor (not C)) xor (not C)
Group 4	(A and B) and (A)
Group 4	(A and B) or (A)
Group 4	(A and B) and (B)
Group 4	(A and B) or (B)
Group 4	(A and B) and (C)
Group 4	(A and B) or (C)
Group 4	(A and B) and (not A)
Group 4	(A and B) or (not A)
Group 4	(A and B) and (not B)
Group 4	(A and B) or (not B)
Group 4	(A and B) and (not C)
Group 4	(A and B) or (not C)
Group 4	(A or B) and (A)
Group 4	(A or B) or (A)
Group 4	(A or B) and (B)
Group 4	(A or B) or (B)
Group 4	(A or B) and (C)
Group 4	(A or B) or (C)

Group 4	(A or B) and (not A)
Group 4	(A or B) or (not A)
Group 4	(A or B) and (not B)
Group 4	(A or B) or (not B)
Group 4	(A or B) and (not C)
Group 4	(A or B) or (not C)
Group 4	(A xor B) and (A)
Group 4	(A xor B) or (A)
Group 4	(A xor B) and (B)
Group 4	(A xor B) or (B)
Group 4	(A xor B) and (C)
Group 4	(A xor B) or (C)
Group 4	(A xor B) and (not A)
Group 4	(A xor B) or (not A)
Group 4	(A xor B) and (not B)
Group 4	(A xor B) or (not B)
Group 4	(A xor B) and (not C)
Group 4	(A xor B) or (not C)
Group 4	(A == B) and (A)
Group 4	(A == B) or (A)
Group 4	(A == B) and (B)
Group 4	(A == B) or (B)
Group 4	(A == B) and (C)
Group 4	(A == B) or (C)
Group 4	(A == B) and (not A)
Group 4	(A == B) or (not A)
Group 4	(A == B) and (not B)
Group 4	(A == B) or (not B)
Group 4	(A == B) and (not C)
Group 4	(A == B) or (not C)
Group 4	(A != B) and (A)
Group 4	(A != B) or (A)
Group 4	(A != B) and (B)
Group 4	(A != B) or (B)
Group 4	(A != B) and (C)
Group 4	(A != B) or (C)
Group 4	(A != B) and (not A)
Group 4	(A != B) or (not A)
Group 4	(A != B) and (not B)
Group 4	(A != B) or (not B)
Group 4	(A != B) and (not C)
Group 4	(A != B) or (not C)
Group 4	(not (A and B)) and (A)
Group 4	(not (A and B)) or (A)
Group 4	(not (A and B)) and (B)
Group 4	(not (A and B)) or (B)
Group 4	(not (A and B)) and (C)

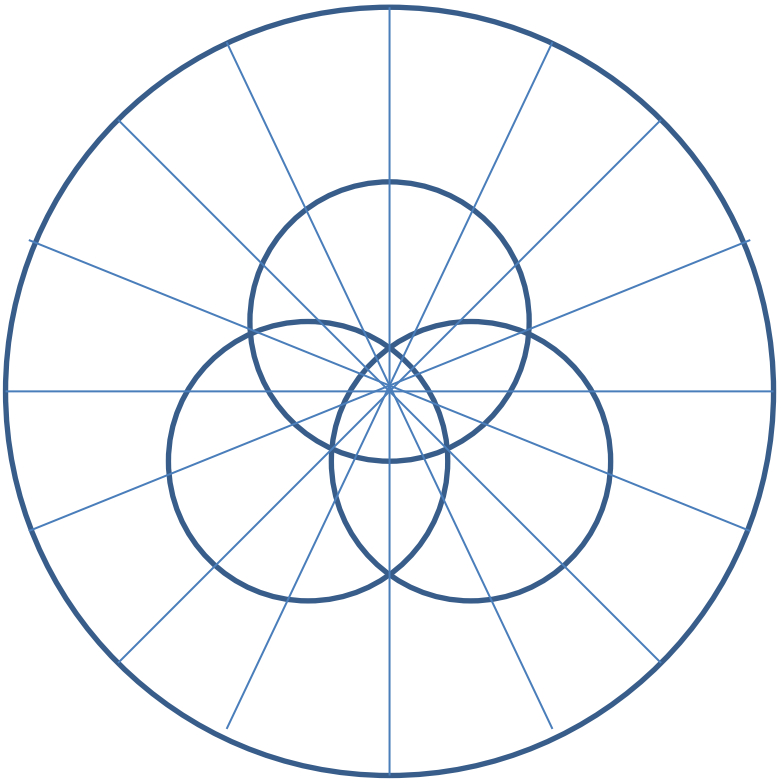
Group 4	(not (A and B)) or (C)
Group 4	(not (A and B)) and (not A)
Group 4	(not (A and B)) or (not A)
Group 4	(not (A and B)) and (not B)
Group 4	(not (A and B)) or (not B)
Group 4	(not (A and B)) and (not C)
Group 4	(not (A and B)) or (not C)
Group 5	not (A and B or A and B)
Group 5	not (A and B and A and B)
Group 5	not (A and B or A or B)
Group 5	not (A and B and A or B)
Group 5	not (A and B or A xor B)
Group 5	not (A and B and A xor B)
Group 5	not (A and B or A == B)
Group 5	not (A and B and A == B)
Group 5	not (A and B or A != B)
Group 5	not (A and B and A != B)
Group 5	not (A and B or not (A and B))
Group 5	not (A and B and not (A and B))
Group 5	not (A and B or not (A or B))
Group 5	not (A and B and not (A or B))
Group 5	not (A and B or not (A xor B))
Group 5	not (A and B and not (A xor B))
Group 5	not (A and B or not (A == B))
Group 5	not (A and B and not (A == B))
Group 5	not (A and B or not (A != B))
Group 5	not (A and B and not (A != B))
Group 5	not (A or B or A and B)
Group 5	not (A or B and A and B)
Group 5	not (A or B or A or B)
Group 5	not (A or B and A or B)
Group 5	not (A or B or A xor B)
Group 5	not (A or B and A xor B)
Group 5	not (A or B or A == B)
Group 5	not (A or B and A == B)
Group 5	not (A or B or A != B)
Group 5	not (A or B and A != B)
Group 5	not (A or B or not (A and B))
Group 5	not (A or B and not (A and B))
Group 5	not (A or B or not (A or B))
Group 5	not (A or B and not (A or B))
Group 5	not (A or B or not (A xor B))
Group 5	not (A or B and not (A xor B))
Group 5	not (A or B or not (A == B))
Group 5	not (A or B and not (A == B))
Group 5	not (A or B or not (A != B))
Group 5	not (A or B and not (A != B))

Group 5	not (A xor B or A and B)
Group 5	not (A xor B and A and B)
Group 5	not (A xor B or A or B)
Group 5	not (A xor B and A or B)
Group 5	not (A xor B or A xor B)
Group 5	not (A xor B and A xor B)
Group 5	not (A xor B or A == B)
Group 5	not (A xor B and A == B)
Group 5	not (A xor B or A != B)
Group 5	not (A xor B and A != B)
Group 5	not (A xor B or not (A and B))
Group 5	not (A xor B and not (A and B))
Group 5	not (A xor B or not (A or B))
Group 5	not (A xor B and not (A or B))
Group 5	not (A xor B or not (A xor B))
Group 5	not (A xor B and not (A xor B))
Group 5	not (A xor B or not (A == B))
Group 5	not (A xor B and not (A == B))
Group 5	not (A xor B or not (A != B))
Group 5	not (A xor B and not (A != B))
Group 5	not (A == B or A and B)
Group 5	not (A == B and A and B)
Group 5	not (A == B or A or B)
Group 5	not (A == B and A or B)
Group 5	not (A == B or A xor B)
Group 5	not (A == B and A xor B)
Group 5	not (A == B or A == B)
Group 5	not (A == B and A == B)
Group 5	not (A == B or A != B)
Group 5	not (A == B and A != B)
Group 5	not (A == B or not (A and B))
Group 5	not (A == B and not (A and B))
Group 6	((A and B) xor (A)) and (A)
Group 6	((A and B) == (A)) or not (A)
Group 6	((A and B) xor (B)) and (B)
Group 6	((A and B) == (B)) or not (B)
Group 6	((A and B) xor (C)) and (C)
Group 6	((A and B) == (C)) or not (C)
Group 6	((A and B) xor (not A)) and (not A)
Group 6	((A and B) == (not A)) or not (not A)
Group 6	((A and B) xor (not B)) and (not B)
Group 6	((A and B) == (not B)) or not (not B)
Group 6	((A and B) xor (not C)) and (not C)
Group 6	((A and B) == (not C)) or not (not C)
Group 6	((A or B) xor (A)) and (A)
Group 6	((A or B) == (A)) or not (A)
Group 6	((A or B) xor (B)) and (B)

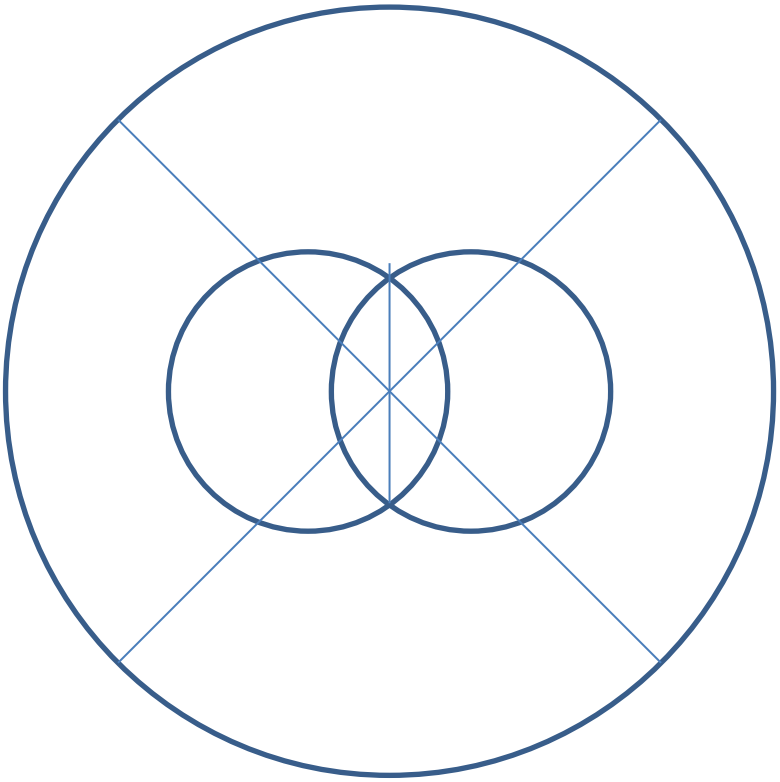
Group 6 ((A or B) == (B)) or not (B)
Group 6 ((A or B) xor (C)) and (C)
Group 6 ((A or B) == (C)) or not (C)
Group 6 ((A or B) xor (not A)) and (not A)
Group 6 ((A or B) == (not A)) or not (not A)
Group 6 ((A or B) xor (not B)) and (not B)
Group 6 ((A or B) == (not B)) or not (not B)
Group 6 ((A or B) xor (not C)) and (not C)
Group 6 ((A or B) == (not C)) or not (not C)
Group 6 ((A xor B) xor (A)) and (A)
Group 6 ((A xor B) == (A)) or not (A)
Group 6 ((A xor B) xor (B)) and (B)
Group 6 ((A xor B) == (B)) or not (B)
Group 6 ((A xor B) xor (C)) and (C)
Group 6 ((A xor B) == (C)) or not (C)
Group 6 ((A xor B) xor (not A)) and (not A)
Group 6 ((A xor B) == (not A)) or not (not A)
Group 6 ((A xor B) xor (not B)) and (not B)
Group 6 ((A xor B) == (not B)) or not (not B)
Group 6 ((A xor B) xor (not C)) and (not C)
Group 6 ((A xor B) == (not C)) or not (not C)
Group 6 ((A == B) xor (A)) and (A)
Group 6 ((A == B) == (A)) or not (A)
Group 6 ((A == B) xor (B)) and (B)
Group 6 ((A == B) == (B)) or not (B)
Group 6 ((A == B) xor (C)) and (C)
Group 6 ((A == B) == (C)) or not (C)
Group 6 ((A == B) xor (not A)) and (not A)
Group 6 ((A == B) == (not A)) or not (not A)
Group 6 ((A == B) xor (not B)) and (not B)
Group 6 ((A == B) == (not B)) or not (not B)
Group 6 ((A == B) xor (not C)) and (not C)
Group 6 ((A == B) == (not C)) or not (not C)
Group 6 ((A != B) xor (A)) and (A)
Group 6 ((A != B) == (A)) or not (A)
Group 6 ((A != B) xor (B)) and (B)
Group 6 ((A != B) == (B)) or not (B)
Group 6 ((A != B) xor (C)) and (C)
Group 6 ((A != B) == (C)) or not (C)
Group 6 ((A != B) xor (not A)) and (not A)
Group 6 ((A != B) == (not A)) or not (not A)
Group 6 ((A != B) xor (not B)) and (not B)
Group 6 ((A != B) == (not B)) or not (not B)
Group 6 ((A != B) xor (not C)) and (not C)
Group 6 ((A != B) == (not C)) or not (not C)
Group 6 ((not (A and B)) xor (A)) and (A)
Group 6 ((not (A and B)) == (A)) or not (A)

Group 6 ((not (A and B)) xor (B)) and (B)
Group 6 ((not (A and B)) == (B)) or not (B)
Group 6 ((not (A and B)) xor (C)) and (C)
Group 6 ((not (A and B)) == (C)) or not (C)
Group 6 ((not (A and B)) xor (not A)) and (not A)
Group 6 ((not (A and B)) == (not A)) or not (not A)
Group 6 ((not (A and B)) xor (not B)) and (not B)
Group 6 ((not (A and B)) == (not B)) or not (not B)
Group 6 ((not (A and B)) xor (not C)) and (not C)
Group 6 ((not (A and B)) == (not C)) or not (not C)

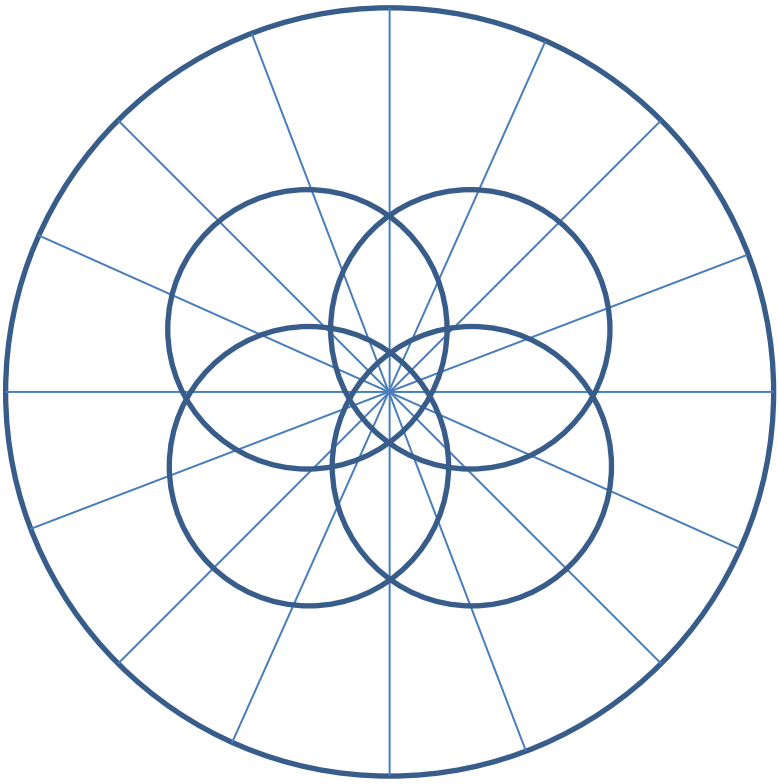
A=4 x B=4 x C=4 = 64



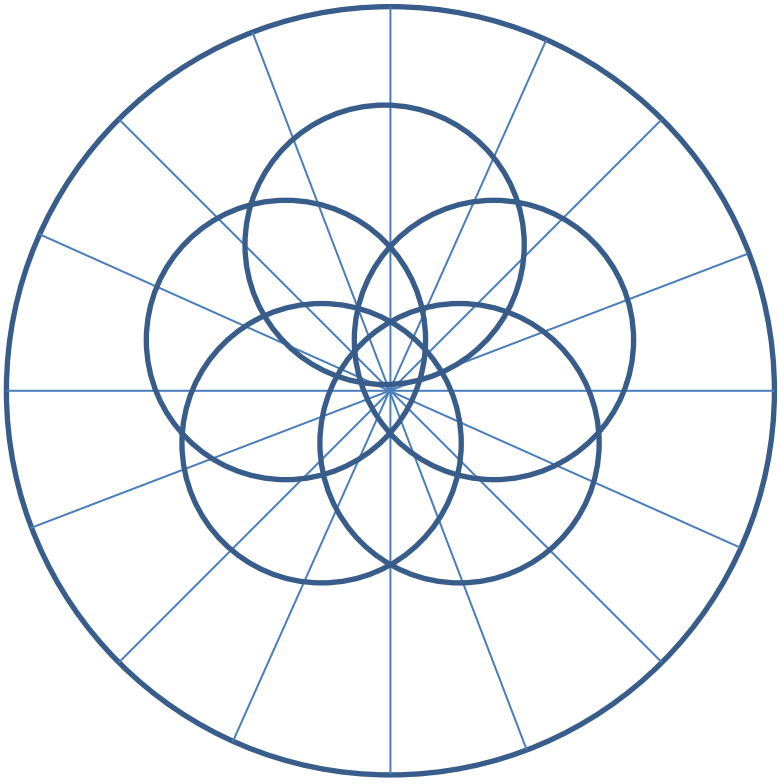
A=4 x B=4 = 16



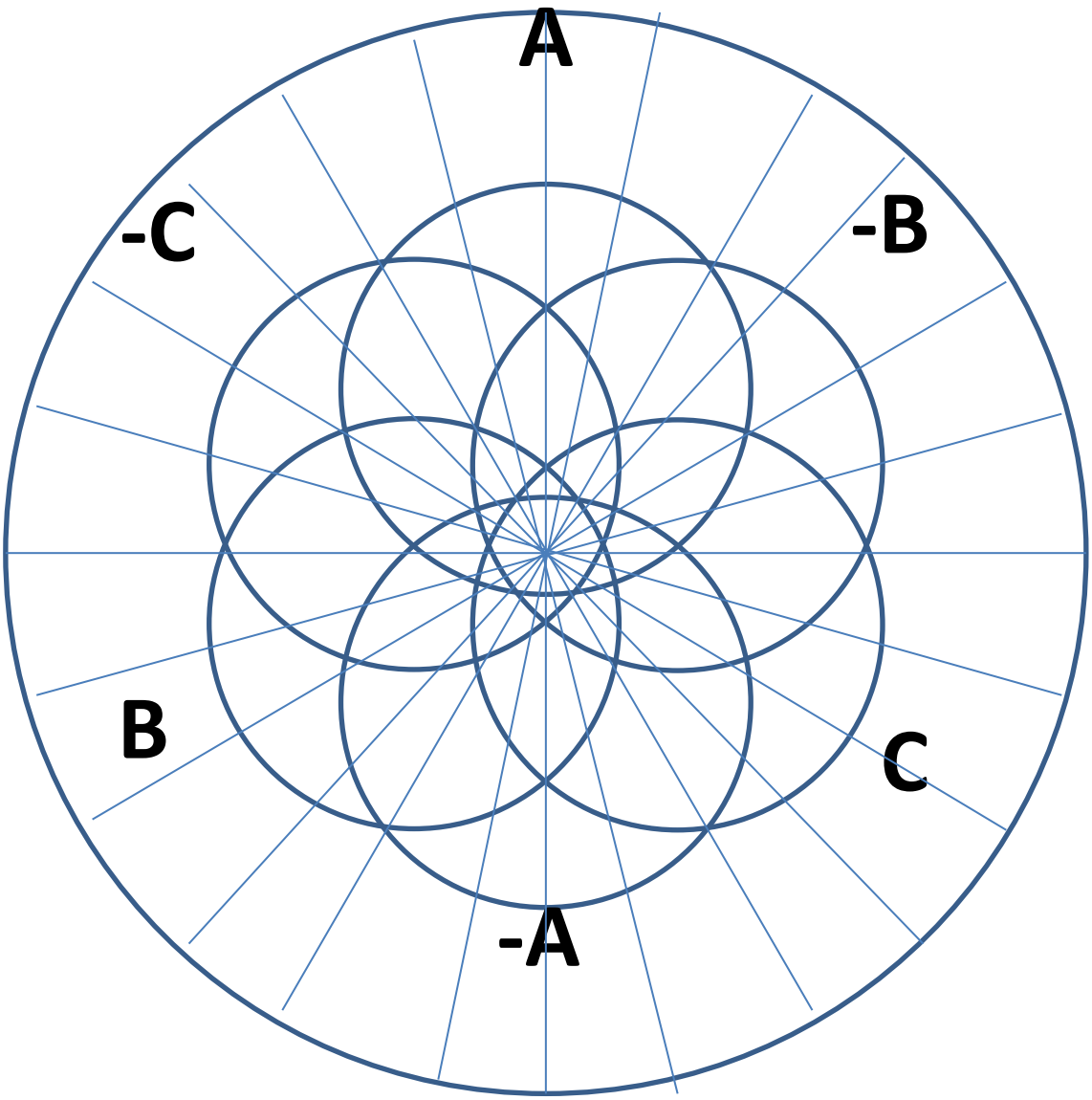
A=4 x B=4 x C=4 x D=4 = 256



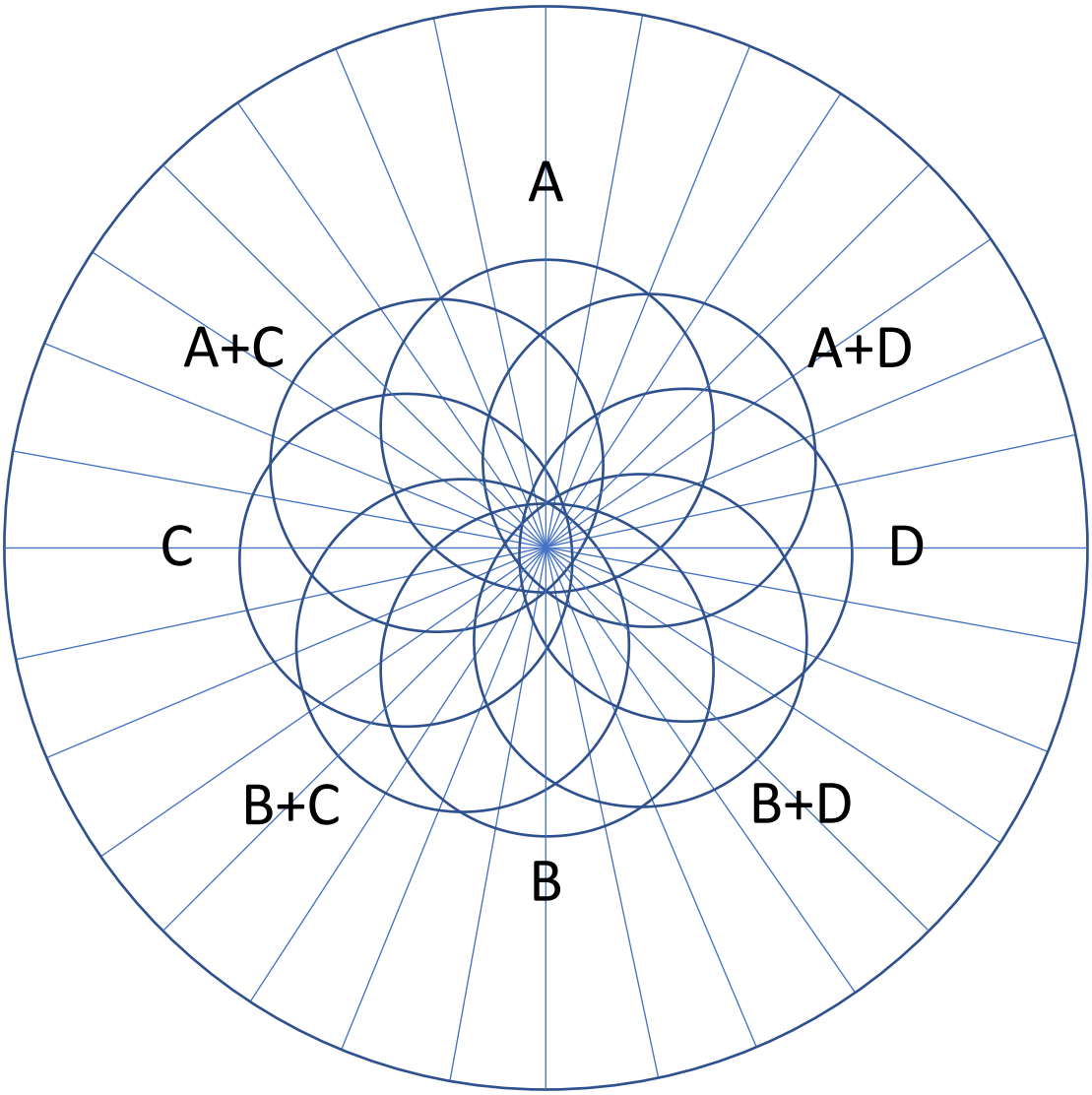
A=4 x B=4 x C=4 x D=4 x E=4
= 1024



A=4 x B=4 x C=4 x D=4 x E=4 x F=4 = 4096

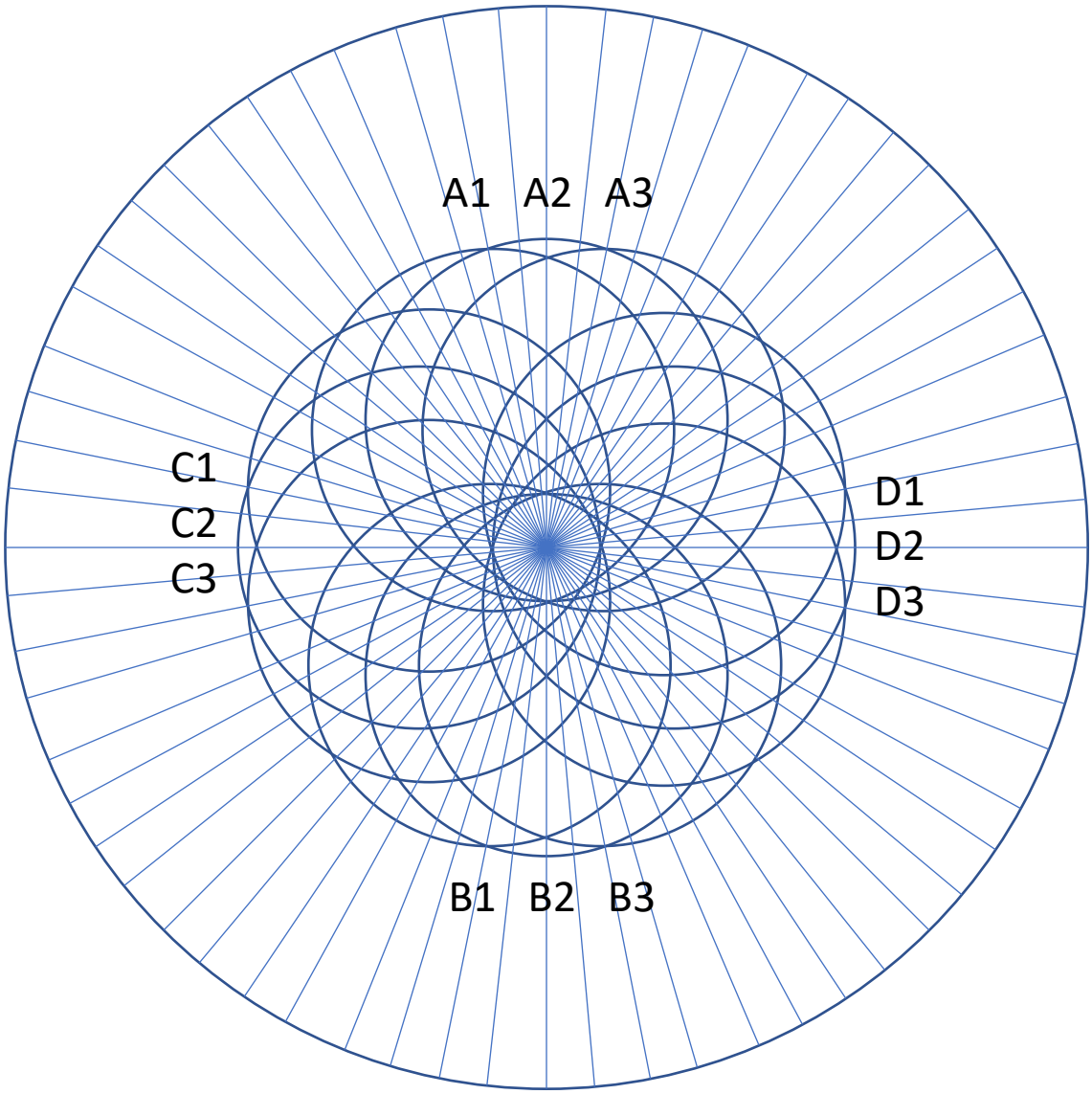


4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 = 65 536



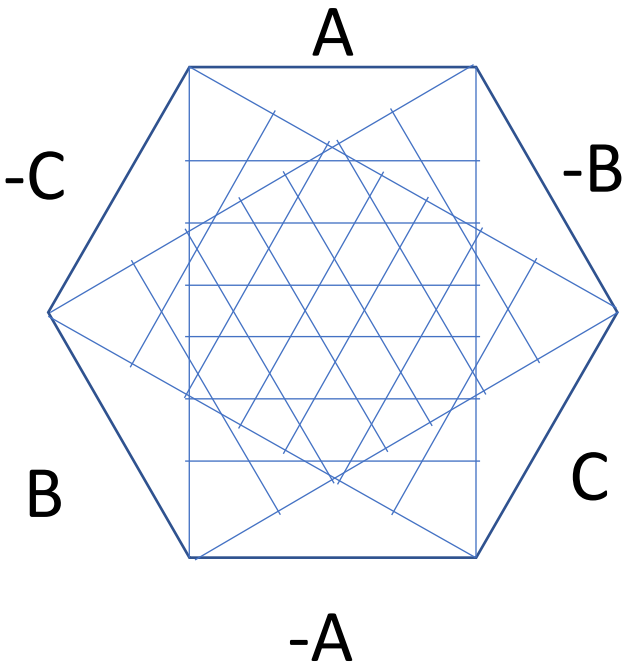
4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 x 4 =
16 777 216

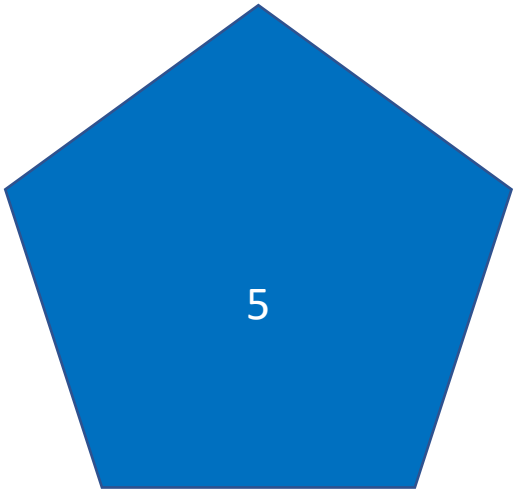
The next one is 20 laps.

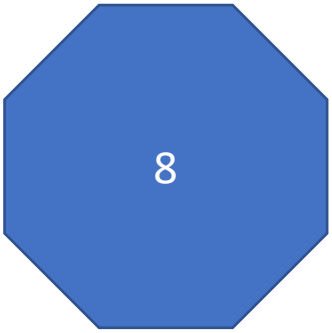
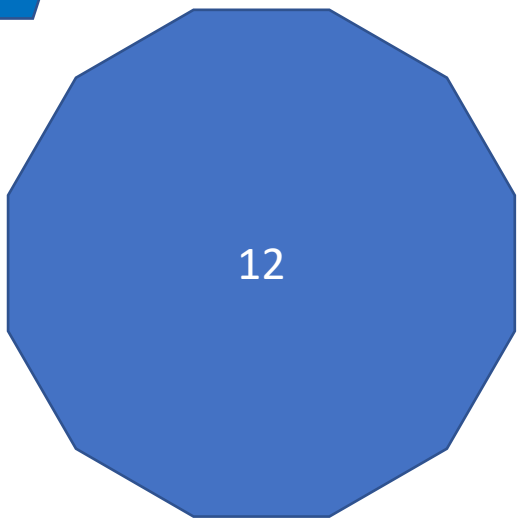


Supercomputer in your notebook (requires development of rules, procedures, algorithms): 6 circles: 24 divisions outside the flower and in the center, 30 petals divided into 4 parts (a+ = 5 * 4, b+ = 5 * 4, c+ = 5 * 4, -a+ = 5 * 4, -b+ = 5 * 4, -c+ = 5 * 4) ...

A possible formula could combine these states logically, e.g.,
F=(A1∧A2∧A3)∨(B1∧B2∧B3)∨(C1∧C2∧C3)∨(D1∧D2∧D3)







Calculation tables